CAN Bus Simulator

User Manual

Table of Contents

1 Features Highlights

- Support CAN BUS Protocol, J1939 Protocol and CANopen Server Protocol
- Support Matlab/Simulink in Windows/Linux/MacOS Platform
- Support Windows 7/8/10/11 and above.
- Support as many as 4 Devices to simulate.
- Support CAN BUS 2.0A and 2.0B.
- Support Remote Frame transmitting and receiving.
- Monitor all CAN Bus activities inside simulated device
- Graphic state machine to display simulated devices
- Simulator software can be called by customer software and Customer software can easily send simulated parameters in real time
- CAN Bus baud rate is some values from 25K to 1M bps
- 3 KV Isolation between CAN Bus and PC
- Power by external supply range is 5.5V to 28VDC or/and USB Connection.

***Notes**: Alternatively, if you have Matlab/Simulink License, you can use Simulink library and create your simulator directly, just read our Matlab/Simulink datasheet.*

2 Hardware

Fig.1 Hardware of CAN Bus Analyzer/Simulator

If you use it as J1939 Simulator, you can order Deutsch Cable separately. The part number is DFLDC9CV2.

2.1 Pin Assignment

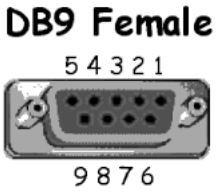


Fig.2 DB9 Pinout

Table 1 Pinout

Pin	Name	Description
1	Reserved	Leave it unconnected.
2	CANL	CAN Bus Signal: CAN Low
3	CAN BUS Signal Ground	CAN Bus Signal and Power ground. 5.5 to 28V DC Power supply input - Side.
4	Reserved	Leave it unconnected.
5	Shield Ground	Connect earth ground or Leave it unconnected.
6	CAN BUS Signal Ground	CAN Bus Signal and Power ground. 5.5 to 28V DC Power supply input - Side.
7	CANH	CAN Bus Signal: CAN High
8	Reserved	Leave it unconnected.
9	5.5 to 28V Power (+)	This is option. 5.5 to 28V DC Power supply input + Side. You can leave it unconnected, just use USB connection Power supply. The power supply negative side is pin6 (CAN BUS Signal Ground).

2.2 LED Indication

There are 4 LEDs to indicate the CAN Bus Analyzer/Simulator status. 2 LEDs' color is Green. 2 LEDs' color is Red

1 Supported Function LED (Red color)

If this LED is bright, it means CAN BUS Simulator. No echo for transmitting CAN Frame.

If this LED is blinking, it means CAN BUS Analyzer. It has echo receiving for transmitting CAN Frame.

2 Enable LED (Red Color)

If this LED is Dark, it means CAN Bus is disabled, It can not send CAN Acknowledge bit

If this LED is bright, it means CAN Bus is enabled, It can send CAN Acknowledge bit

3 CAN TX LED (Green color)

When CAN Bus Analyzer/Simulator transmits any CAN Frame, this LED will be on.

4 **CAN RX LED (Green color)**

When CAN Bus Analyzer/Simulator receives any CAN Frame, this LED will be on.

3 Software Interface

CAN Bus Simulator software has 3 areas: Menu Area, State Machine Tool Area and Status Area as Fig.3 below:

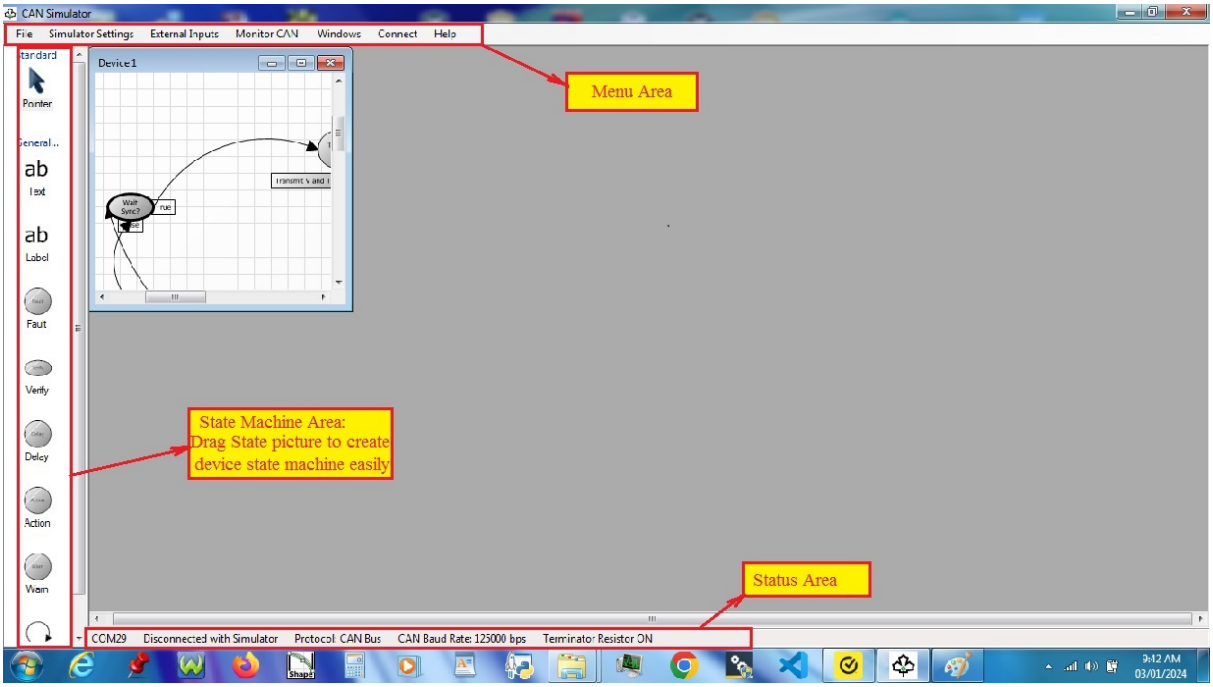


Fig.3 Software Interface

You operate software by click Menu item and drag/drop state machine icon. You will see software Status in "Status bar" such as "COM port number, Disconnected with Simulator/Connected with Simulator (CAN Bus Simulator Stopped / Started), CAN Bus Protocol, CAN baud Rate, and 120 Ohms terminator on/off ".

In the Menu area, there are "File", "Simulator Settings", "External Inputs", "Object Dictionary (Only visible when CANopen is used)", "Monitor CAN", "Windows", "Connect", and "Help" items.

For Menu item "File", it has "New State Machine", "Use Settings File ... ", "Use OD File...", "Save Settings", "Save Settings As ... ", "Save State Machine", "Save State Machine As ... ", "Save OD File", "Save OD File As...", "Save All", "Initial State Lock", and "Exit" as Fig.4 below:

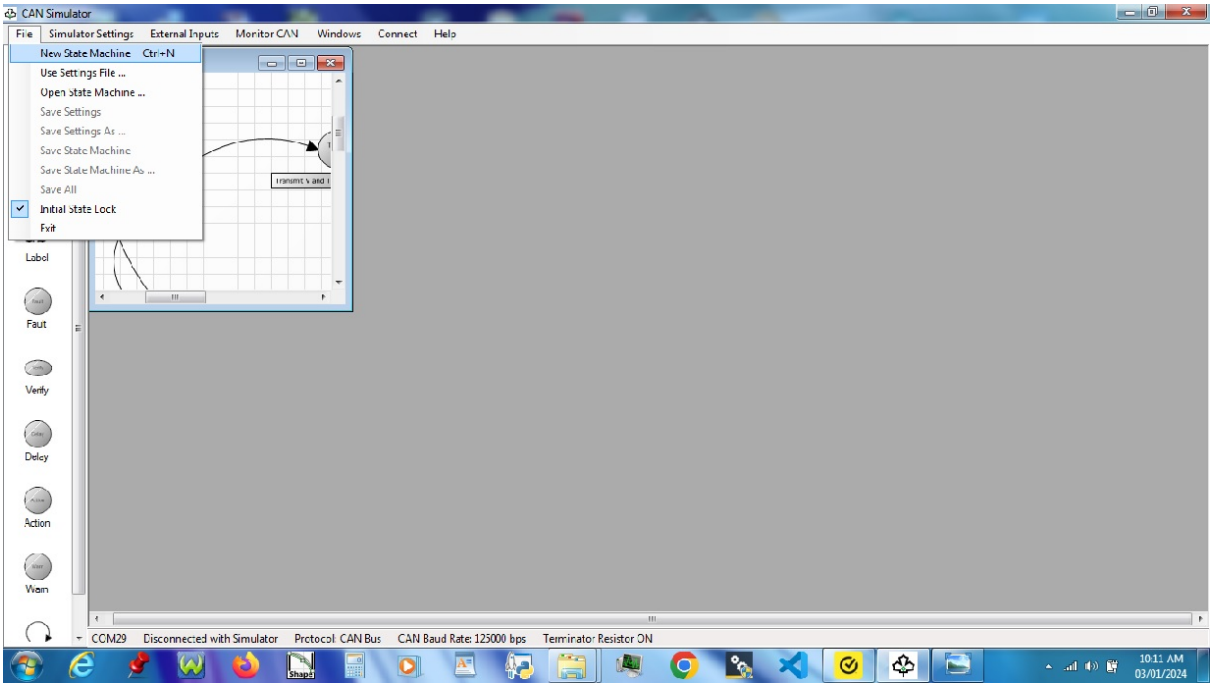


Fig.4 File menu

The item "New State Machine" is used to open new state machine windows, which has empty state machine, for each simulated device. You just drag/drop left-side tool into state machine window.

The item "Use Settings File ... " is loading CAN BUS Simulator settings from PC file. CAN BUS Simulator settings include CAN BUS baud rate, Protocol name, Simulated Device quantities, Analog inputs quantities and relation with raw data, Discrete inputs, Fault and warning signals, CAN Bus filter, Acceptable received CAN ID, Watchdog, Acceptable received PGN, TPDO and RPDO.

The item "Use OD File ... " is loading CANopen Dictionary from PC file. It is only available for CANopen protocol.

The item "Save Settings" is save CAN BUS Simulator settings to PC file. File name is previous name you chose.

The item "Save Settings As ... " is save CAN BUS Simulator settings to PC file. File name is decided by popup dialog.

The item "Save State Machine" is save CAN BUS Simulator State machine to PC file. File name is previous name you chose.

The item "Save State Machine As ... " is save CAN BUS Simulator State machine to PC file. File name is decided by popup dialog.
The item "Save OD File" is save Object Dictionary to PC file. File name is previous name you chose. It is only available for CANopen protocol.
The item "Save OD File As ... " is save Object Dictionary to PC file. File name is decided by popup dialog. It is only available for CANopen protocol.
The item "Save All" is save CAN BUS Simulator Settings, Object Dictionary and State machine to PC file. File name is previous name you chose.
The item "Exit" will exit this application software, all settings will kept automatically for next time running.

For Menu item "Simulator Settings", It will display wizards windows for setting all parameters, as Fig.5 below:

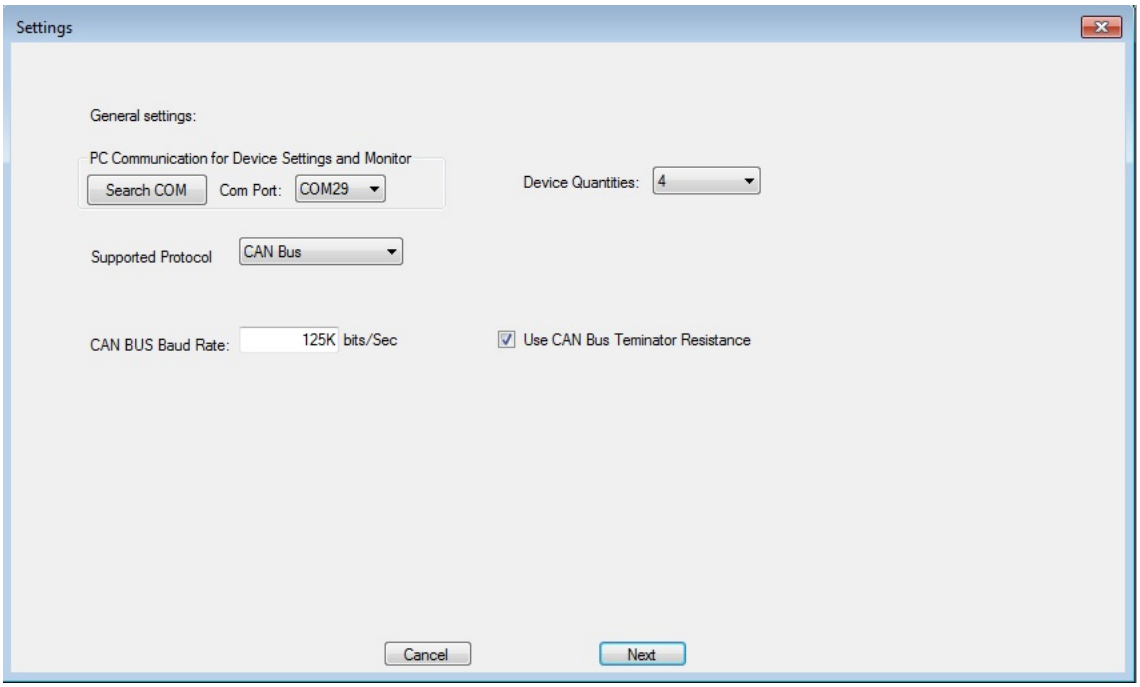


Fig.5 First dialog for "Simulator Settings" menu

We will explain how to set them step by step in next section.
For Menu item "External Inputs", it has "Fault and Digital Inputs" , "Warning Inputs", "Analog Inputs" as Fig.6 below:

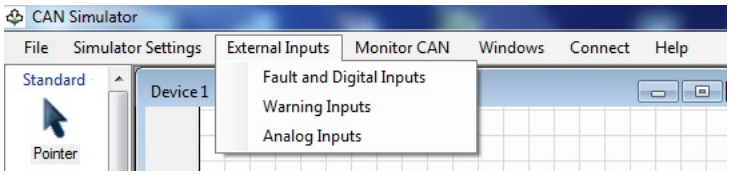


Fig.6 External Inputs menu

The item "Fault and Digital Inputs" means that you will open a sub-window which displays/ sets all faults and discrete inputs.

Please see Fig. 7 for "Fault and Digital Inputs".

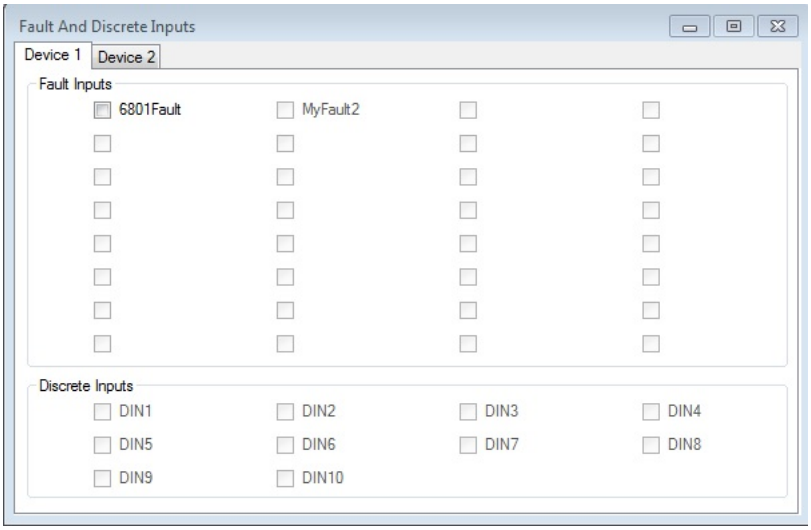


Fig.7 Fault and Digital Inputs

In above Fig.7, there are 2 tabs (Device 1 and Device 2). It may has as many as 4 tabs (Device 1 to Device 4). It depends on how many devices to simulate. If you want view/set other device's faults/ digital inputs, you can click item "Fault and Digital Inputs" again. You choose different tab, you will see all devices, please see Fig. 8 for 2 devices situation.

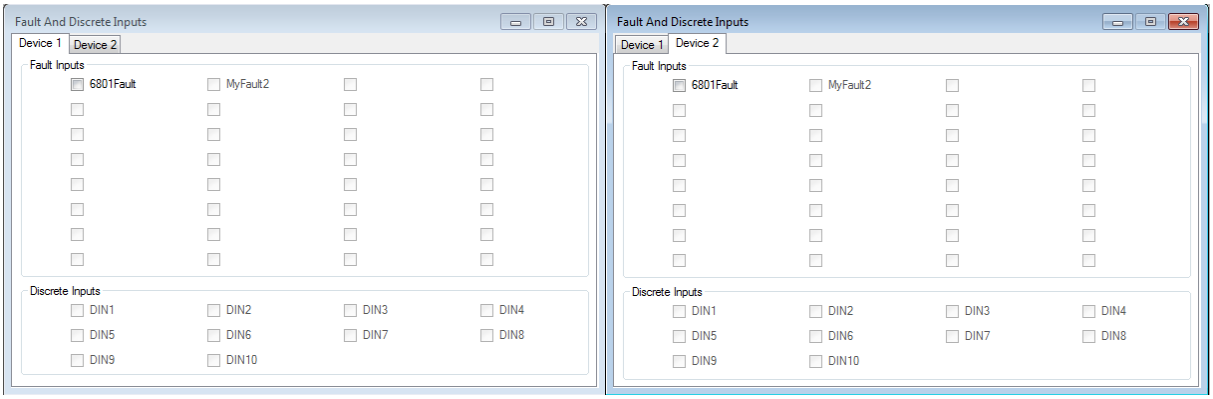


Fig.8 Fault and Digital Inputs for 2 devices

The item "Warning Inputs" means that you will open a sub-window which displays/ sets all warning inputs.
Please see Fig. 9 for "Warning Inputs".

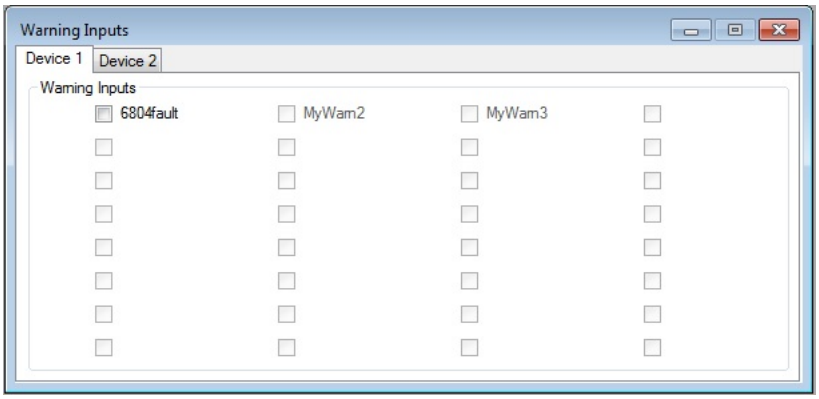


Fig.9 Warning Inputs

Similarly, you can click "Warning Inputs" multiple times to display/set different devices.

The item "Analog Inputs" means that you will open a sub-window which displays/ sets Analog inputs.
Please see Fig. 10 for "Analog Inputs".

Fig.10 Analog Inputs

Similarly, you can click "Analog Inputs" multiple times to display/set different devices.
You can set each analog value one by one. However, sometimes, you like to set many analog inputs as the same value. So right click inside Analog Inputs, it will popup context menu as Fig.11 below

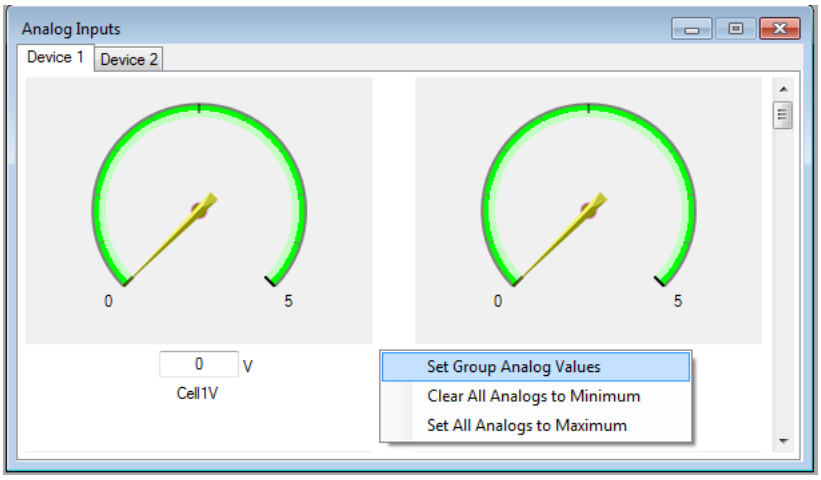


Fig.11 Analog Inputs context menu

Follow up context menu, you can easily set up all analog values to the same value.

For Menu item "Monitor CAN", It will display CAN Bus Activity for simulated Devices. When CAN Bus protocol is selected in settings, "Monitor CAN" window is shown as Fig. 12 below:

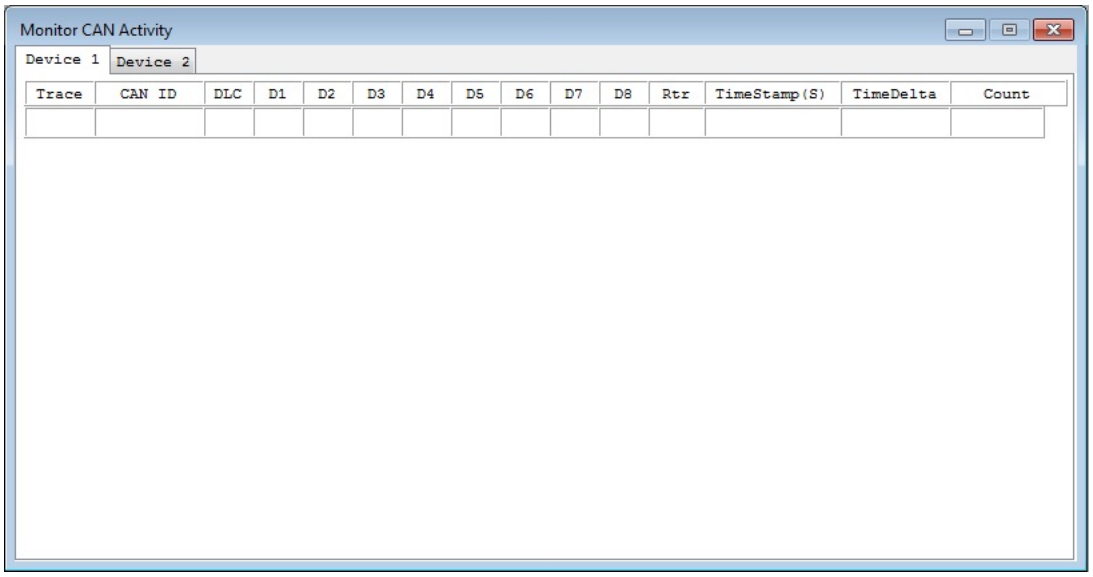


Fig.12 Monitor CAN BUS when it is CAN Bus protocol

When J1939 protocol is selected in settings, "Monitor CAN" window is shown as Fig. 13 below:

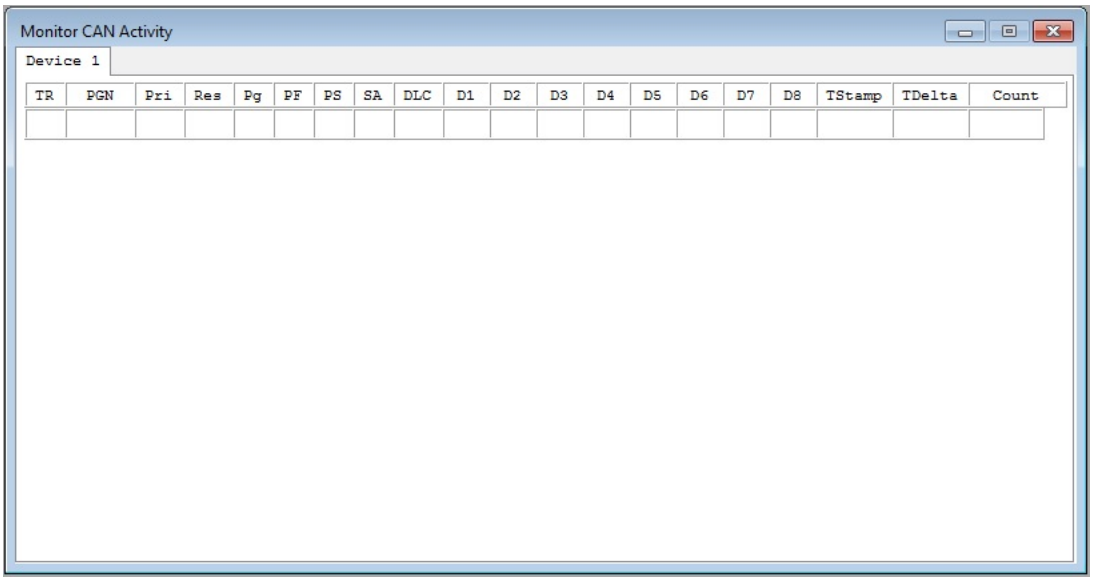


Fig.13 Monitor CAN BUS when it is J1939 protocol

Similarly, you can click "Monitor CAN" multiple times to display Bus activity for different devices.

Notes: *1 For "Transmit/Receive" in Trace (or TR), "TX/RX" is from Simulator viewpoint. Transmit means from Simulator to outside.*

2. Transmitting just means PC send out command to hardware successfully, not means hardware really transmit successfully. So time stamp and delta Time just time related to PC send out command.

3. For transmitting watchdog, Simulator transmits CAN packet automatically. PC didn't send out any command to simulator. So this kind of CAN packet didn't occur in monitor window.

When CANopen protocol is selected in settings, "Monitor CAN" window is shown as Fig. 14. below:

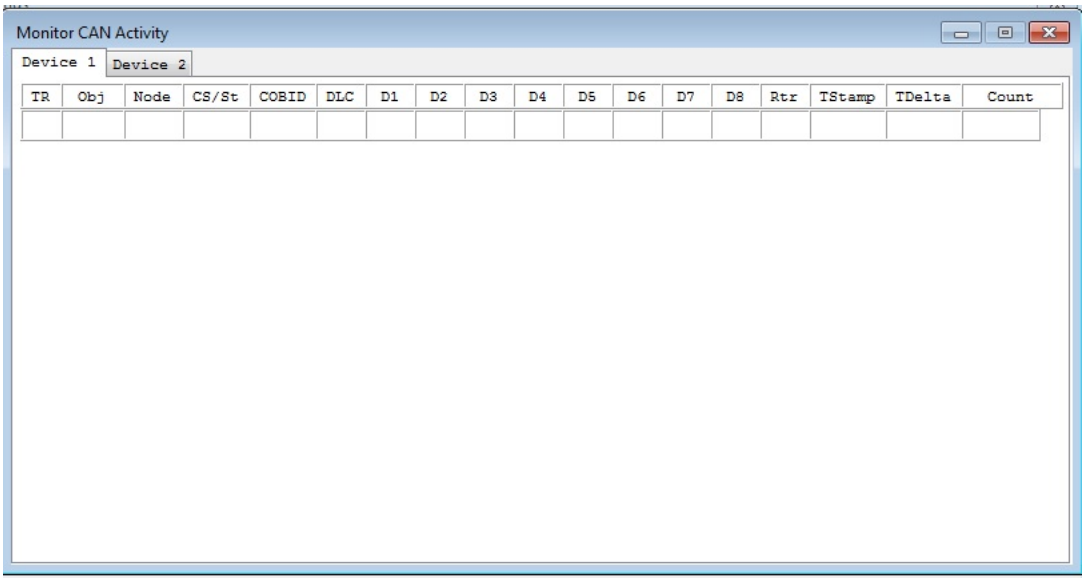


Fig.14 Monitor CANopen when it is CANopen protocol

Notes: *1 CANopen TPDOs are divided into 2 different methods in our simulator; "Hardware implement" and "software implement". The "Hardware implement" means it is transmitted periodically by hardware itself (PC cannot conntrol when to transmit, PC only provides Data values). So* "Monitor CAN" window cannot display "*Hardware implement" TPDO. The "software implement" means PC Control when to transmit and what data values. So "Monitor CAN" window* can display "*Software implement" TPDO. All heartbeat frames are "Hardware implement". You cannot monitor " heartbeat frames" by "Monitor CAN" window. You must use another CAN BUS Analyzer to monitor them.*

2. In "Simulator Settings", you can specify whether TPDO is "Hardware implement"

If CANopen protocol is selected, there is an "Object Dictionary" menu. We will explain "Object Dictionary" in the section of "One Example for CANopen"

For Menu item "Windows", it has "Debug", "Cascade", "Horizontal Title", and "Vertical Title" as Fig.15 below:

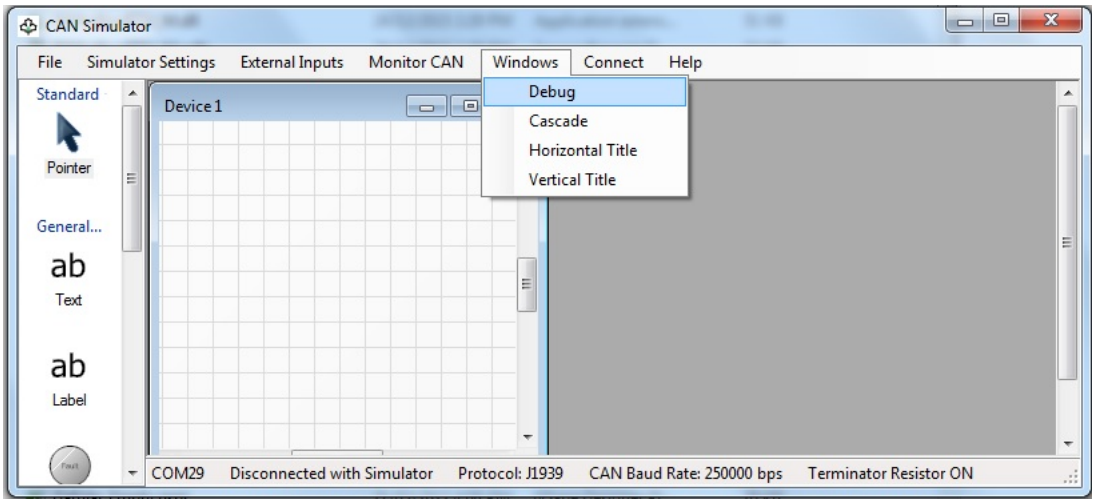


Fig.15 Windows menu

The item "Debug" will display 32 Debugs with unsigned 16 bits of integer in Hex and Decimal format, 32 Debugs with single precision floats, please see Fig. 16

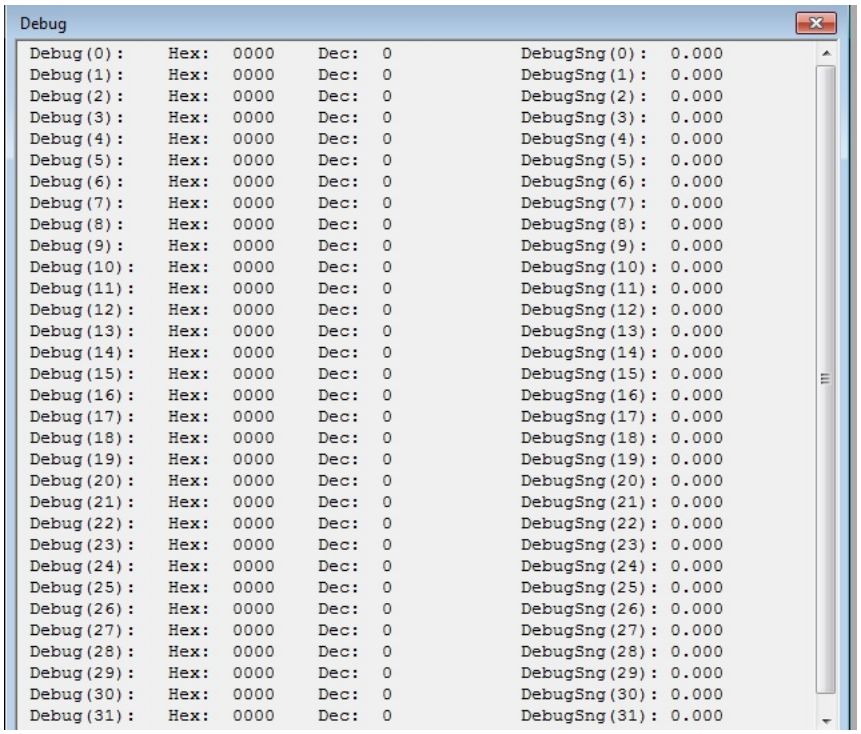


Fig.16 Debug Window

Notes: *1 We cannot communicate between simulated Devices by CAN Bus/J1939/CANopen*

2. Debug registers values are assigned by .customers vb script. You can use Debug registers to exchange data between simulated Devices

For Menu item "Help", it has "Enable/Disable Virtual Serial Port", "About ...", and "Help me" as Fig.17 below:

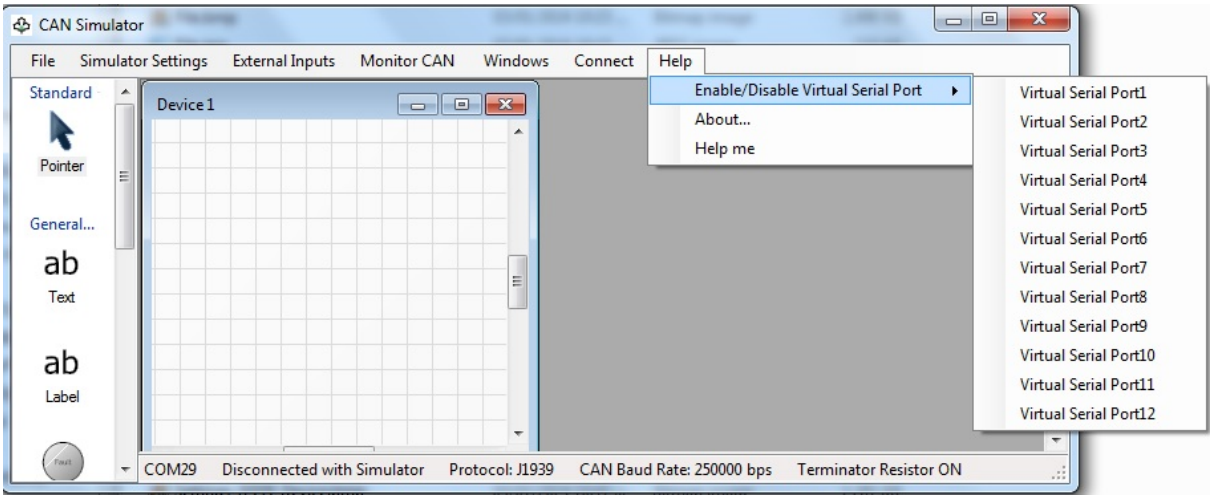


Fig.17 Help Window

The item "Enable/Disable Virtual Serial Port" has 12 sub-items, which are from "Virtual Serial Port1" to "Virtual Serial Port12". IF it is marked as checked, it means this serial port is open. External software can use this Virtual Serial Port to exchange data with simulator. We will tell you how to do it in the last section "Integrated into other software".

4 How to use our software

Firstly, Download software by click: http://dafulaielectronics.com/CANSimulator.zip, and unzip it. Just double click on CANSimulator.exe file to run it. And then follow the steps below:

1 If you are the first time to use this software, the first thing you have to do for new simulator is configuration of "Simulator Settings".

Please click Menu item: " Simulator Settings ". The following dialog occurs:

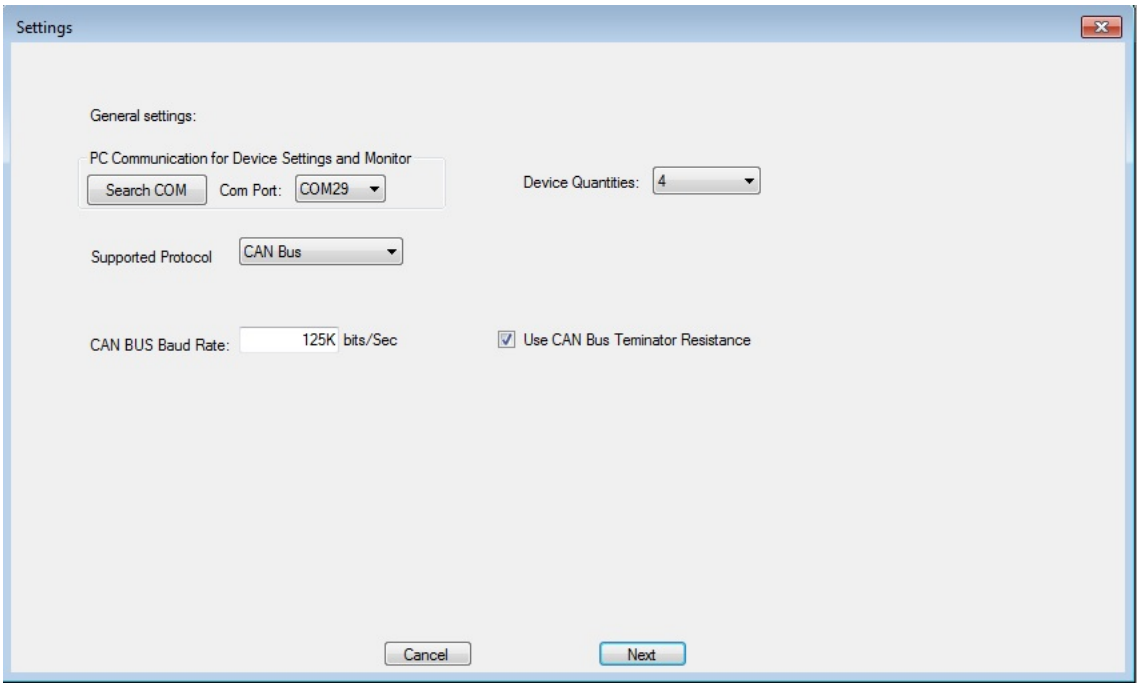


Fig.18 Settings

The button "Search COM" is for searching which USB COM Ports are available for your PC. After click this button, please select correct USB Com port number for your simulator hardware (You can plug /unplug to confirm Simulator COM Port Number).

Device Quantities will be used for how many devices to simulate. It can be selected from 1 to 4. Protocol can be chosen "CAN BUS" or "J1939" or "CANopen" . After all settings are set in above window, please click button "Next", the following window occurs:

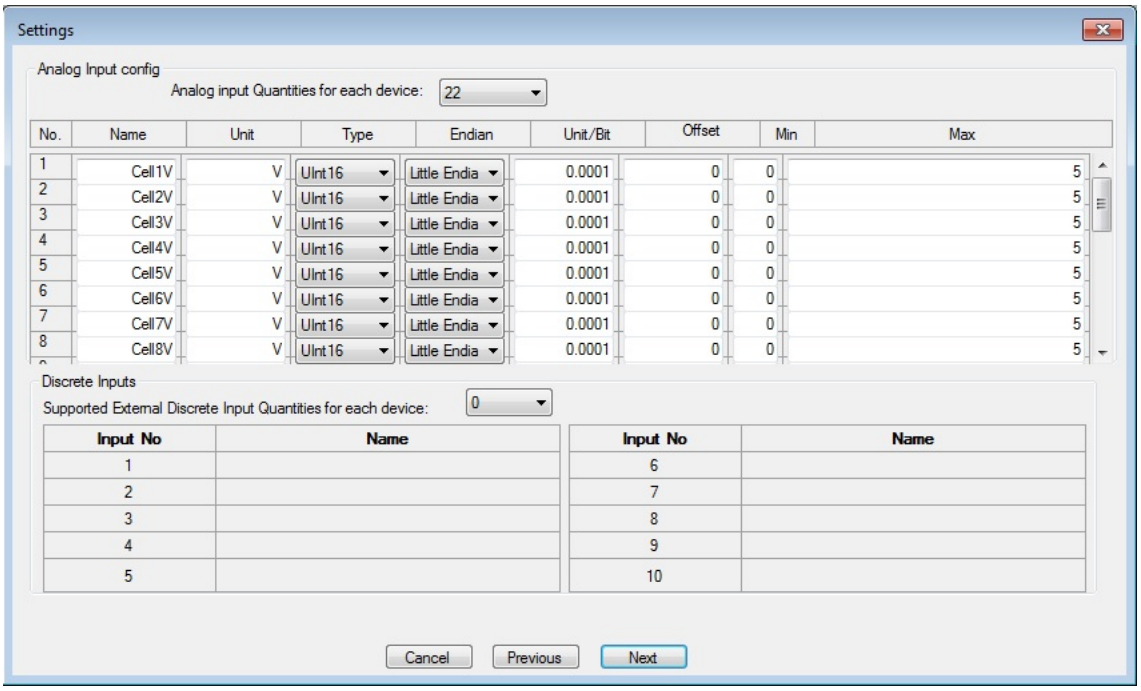


Fig.19 Analog/Discrete Input Settings

In above window, please choose how many analogs you want for each simulated device The Qty is from 0 to 32. All devices you simulated must have the same feature's Analog input, only real-time values are different. For each analog input, it has "Name", "Unit", "Type", "Endian", "Unit/Bit" (that is scale), "Offset", "Min", and "Max" to be set.

Name is just Analog name, it is just for displaying. Unit is analog actual physical unit such as "Meter", "kg" , "V" and "A".

Type is raw data type, it can be "float (4 bytes)", "Int32 (32 bits of signed integer)", "Int16 (16 bits of signed integer)", "Int8 (8 bits of signed integer)", "UInt32 (32 bits of unsigned integer)", "UInt16 (16 bits of unsigned integer)", "UInt8 (8 bits of unsigned integer)"

Endian is for order of bytes. Unit/Bit is scale, it denotes what physical value for raw data. For example, if we use 10 times physical value (such as actual 1 meter, we put into binary raw data as 0xA) to put into binary value, it means 0.1 unit/bit. It is no sense for "float" data because float data is not raw data.

Offset is only available for unsigned integer raw data because we want to use unsigned integer to denote signed data.

Offset must set to what raw data value we must subtract when raw data is zero. Note, it is raw data value , not physical value.

Min is "Minimum physical value". It is for Analog gauge display. Note, it is physical value, not .raw data value.

Max is "Maximum physical value". It is for Analog gauge display. Note, it is physical value, not .raw data value.

For discrete inputs, it is simple, Just select how many digital inputs you want for each simulated device. The Qty is from 0 to 10. All devices you simulated must have the same feature digital inputs, only real-time values are different. Name is just Discrete Input name, it is just for displaying.
After all settings are set in above window, please click button "Next", the following window occurs if your protocol is CAN Bus:

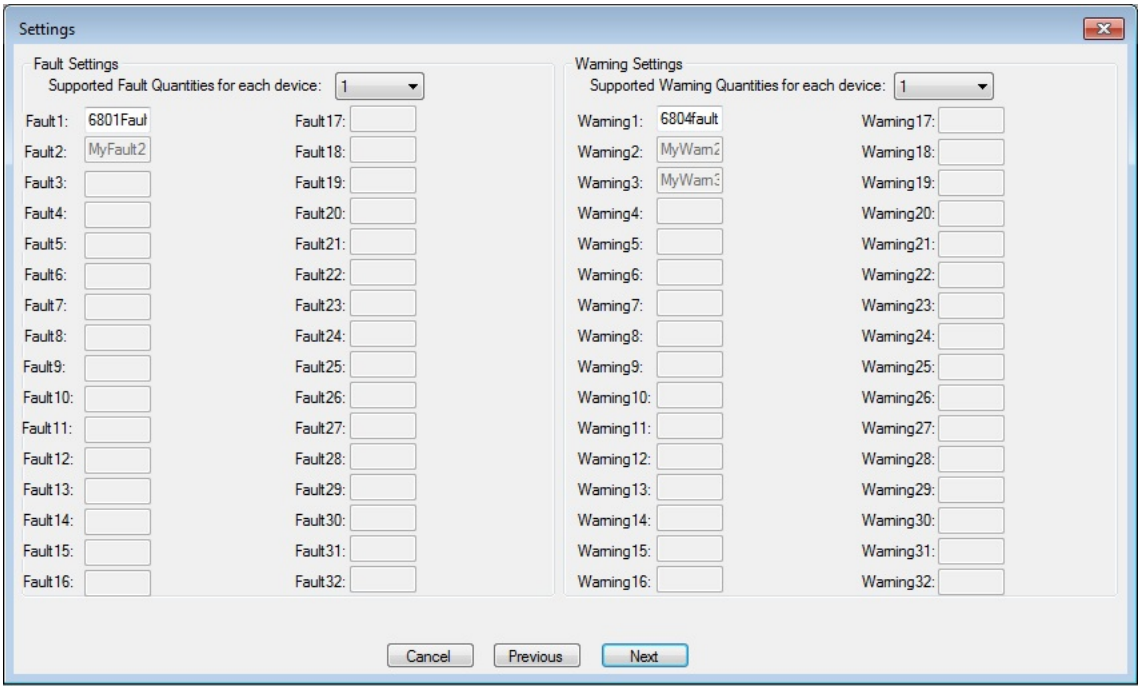


Fig.20 Fault and Warning Settings for CAN BUS Protocol

This is Fault and warnings settings. The fault quantities are 0 to 32, The warning quantities are 0 to 32. Fault and Warnings are no difference. How to handle them depends on your script in state machine.
You don't need draw connection line for how fault or waning State to enter. It enters "State" or "Warning" State automatically as soon as any of fault or warning occurs.
The faults and warnings are special, they are inputs, but outputs too (bi-direction). In your script, you can set analog over/under some thresholds and assign some faults or warnings to true (just like output). Of cause, you can directly use Graphic real-time fault/warning window to set it to true or false (just like input).

For CAN BUS protocol, Fault and warning only need name to be set, it is only for displaying.

However, It is different for J1939 and CANopen protocol. When J1939 Protocol is used, you must set 32 bits of DTC for each fault or warning in addition to name.
The following window occurs if your protocol is J1939:

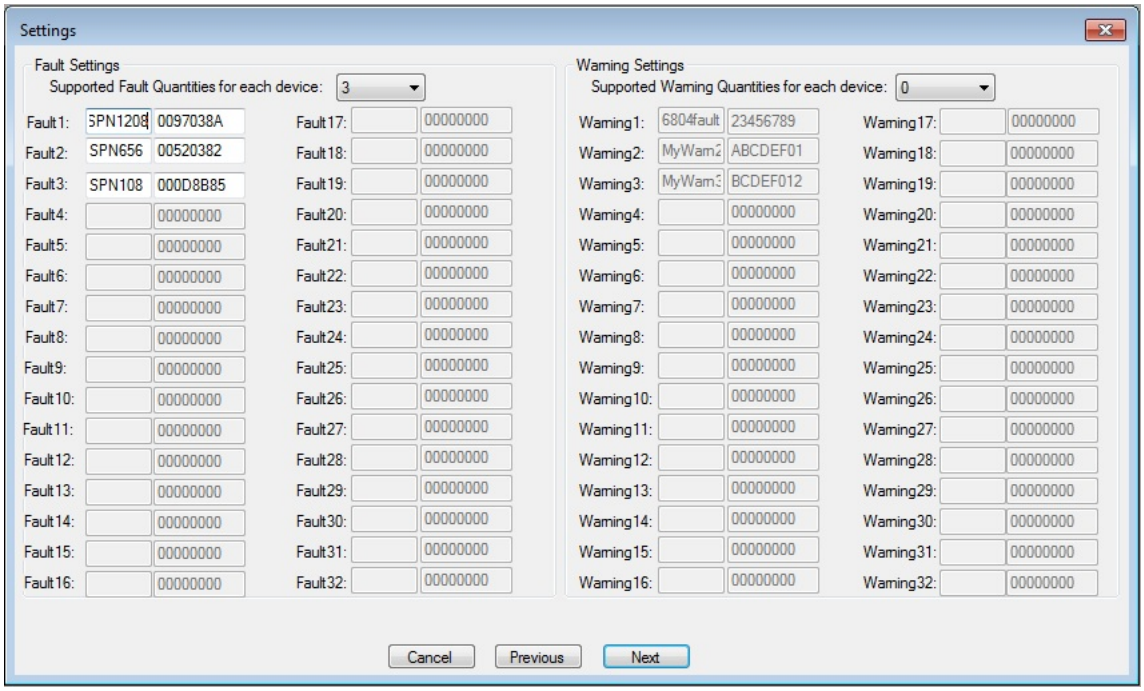


Fig.21 Fault and Warning Settings for J1939 Protocol

You can see the first column you typed is Fault name, the second column you typed is Hex DTC. For J1939 DTC Format, it has 4 versions, version1 to version4. You must set it according to your DTC Format version.

For CANopen protocol, you must set 32 bits of " Error Code" (16 bits) plus "additional information" (16 bits), and 8 bits of "Error Register" for each fault or warning in additional to name.
The first Column is "Fault/Warning Name". The second Column is combination of "additional information" and "Error Code" (Error code is the least significant word, additional information is the most significant word). The third Column is 8 bits of Error Register. Please see Fig.67. Error register has special meaning, you just double click this column to popup dialog window and set it. Please see Fig. 68.

After all settings are set in above window, please click button "Next", the following window occurs if your protocol is J1939:

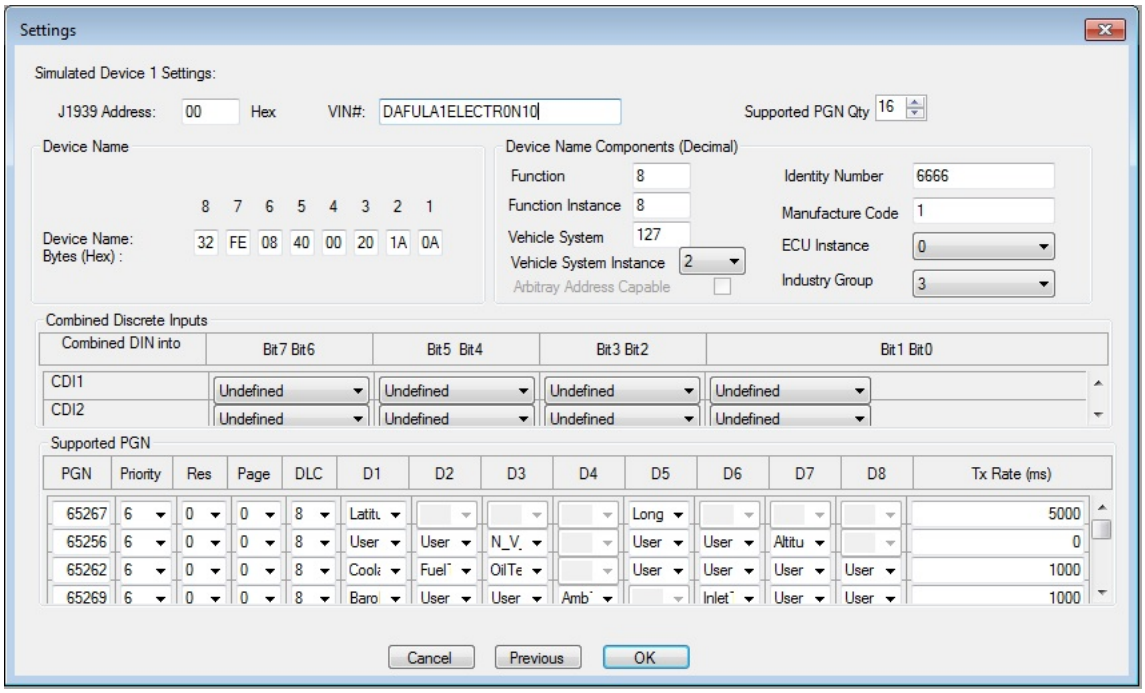


Fig.22 Device Settings for J1939 Protocol

This is Setting for J1939 Device. Most of items are J1939 Standard items. We don't explain J1939 Standard items.

Supported PGN Quantities are 1 to 29. Note, here, Supported PGN Quantities do not include some standard PGN such as VIN/TP/DM1/DM11/PGN59904

Combined Discrete inputs (CDI1 to CDI19) are byte which combine discrete bit inputs (occupy 2 bits each digital input) into one byte.

And column "D1 to D8" of Supported PGN can use CDI1 to CDI10 as data byte.

The Column "Tx Rate (ms)" of Supported PGN is broadcast period in ms. If it is 0, this PGN will be send out only when getting Request from PGN59904

If Device Qty is 1, OK button occurs. Click "OK" to finish all settings. If Device Qty is 2 or 3 or 4, "Next" button occurs. Click "Next" to do the same job as this window.

Notes: *For PGN Data field drop list, the width of item is too small. Sometimes, you cannot see the entire item name. The item list order from top to bottom is "User Defined", "CDI1", "CDI2", "CDI3", "CDI4", "CDI5", "CDI6", "CDI7", "CDI8", "CDI9", "CDI10", "AnalogInput1", "AnalogInput2", ...*

If our protocol is CAN Bus, the above window cannot occur, and the following window will occur:

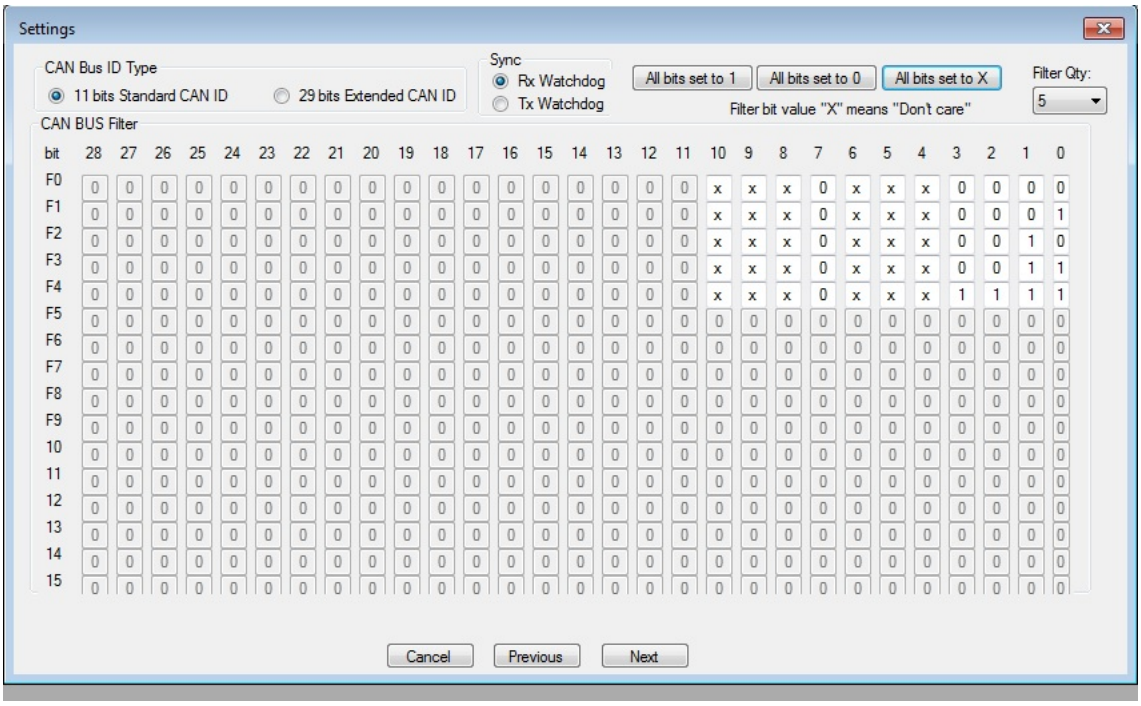


Fig.23 Filter Settings for CAN bus Protocol

This is hardware filter settings. You should choose 11 bits of standard CAN ID or 29 bits of extended CAN ID. Synchronous function is enabled (TxWatchdog or RxWatchdog) or disabled.
Here Synchronous function means that Simulator only transmits CAN packets when it receives or transmits a special CAN packet which is called Synchronous packet.
Synchronous packet only use a special CAN packet which is called Watchdog. If this special CAN packet is transmitted packet, we call it Tx Watchdog, otherwise call it Rx Watchdog.
We will explain Watchdog in next settings for CAN BUS Device.
Totally , hardware can have maximum 16 filters from F0 to F15. If bit value is 1 (0) of Filter, it means that we will accept the CAN ID with the value 1 (0) in the same bit location of filter.
If bit value is x of Filter, it means that we will accept the CAN ID with the any value (1 and 0) in the same bit location of filter.
Above all settings are for all devices you simulate. They must be the same.

***Notes**: Here filters are hardware filters, it will throw away all unwanted CAN Packets. However, for each device we simulate, it has supported CAN ID which is software filter. In general, Hardware filter will accept more CAN packets than each device's software filter.*

After all settings are set in above window for CAN bus protocol, please click button "Next", the following window occurs

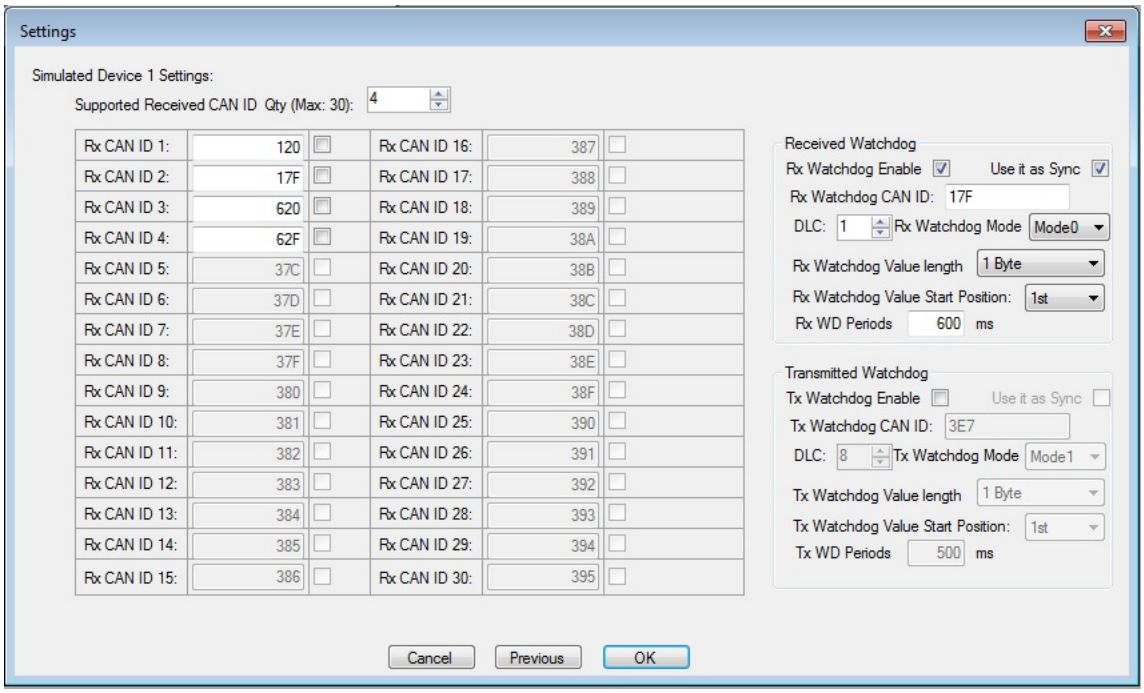


Fig.24 Device Settings for CAN bus Protocol

You can choose "supported received CAN ID" Qty 1 to maximum 30. All received CAN IDs must be allowable by Fig 23 hardware filter. Otherwise, it makes no sense.

You can enable/disable "Received watchdog" (one special CAN packet). The "Received watchdog" is used in my node's CAN BUS communication Error detection and/or "Synchronous function".

How does Watchdog work? We introduce RxWatchdog (Received Watchdog) firstly. Each Device you simulate has one watchdog. Watchdog is ms count. And count can be Up/Down count and it is unsigned. Count length can be configurable by "Rx Watchdog Value length" (bytes quantity). Counter direction (up/down) is decided by watchdog mode. Each watchdog has 3 different working mode.

Mode 0 : Watchdog initial value is 0. Watchdog is free running up count each ms. Watchdog value will return 0 if RxWatchdog value is changed. When count is over periods, it will keep the count value and communication fault will occur (Variable ErrorCom will be true in state machine script).

Mode 1 :Watchdog initial value is 0. Watchdog is free running up-count each ms. When it reaches periods value, it will keep it, and one communication fault will occur. RxWatchdog Data packet can change count value to avoid communication fault.

Mode 2 :Watchdog initial value is equal to periods. Watchdog is free running down count each ms. When it arrive at 0, it will keep 0 and communication fault will occur. RxWatchdog Data

packet can change count value to avoid communication fault.

You can enable/disable "Transmitted watchdog" (one special CAN packet). The "Transmitted watchdog" is used in other node's CAN BUS communication Error detection and/or "Synchronous function".
It has 4 different working mode. However, in this simulator, we only use Mode1/Mode2/Mode3

Mode 1 : TxWatchdog value increases 1 every TxWatchdog's period automatically. When value > maximum unsigned number, it will return 0, and increase again next time
Mode 2 : TxWatchdog value decreases 1 every TxWatchdog's period automatically. When value = 0, it will return maximum unsigned number, and decrease again next time.
Mode 3 : TxWatchdog will have no any value, its data packet length is zero. It is used for CANOpen Sync frame.

All 4 devices you simulated must use the same Synchronous CAN packet if this device you simulated uses "Synchronous function".

If our protocol is CANopen, the above window cannot occur, and the following window will occur:

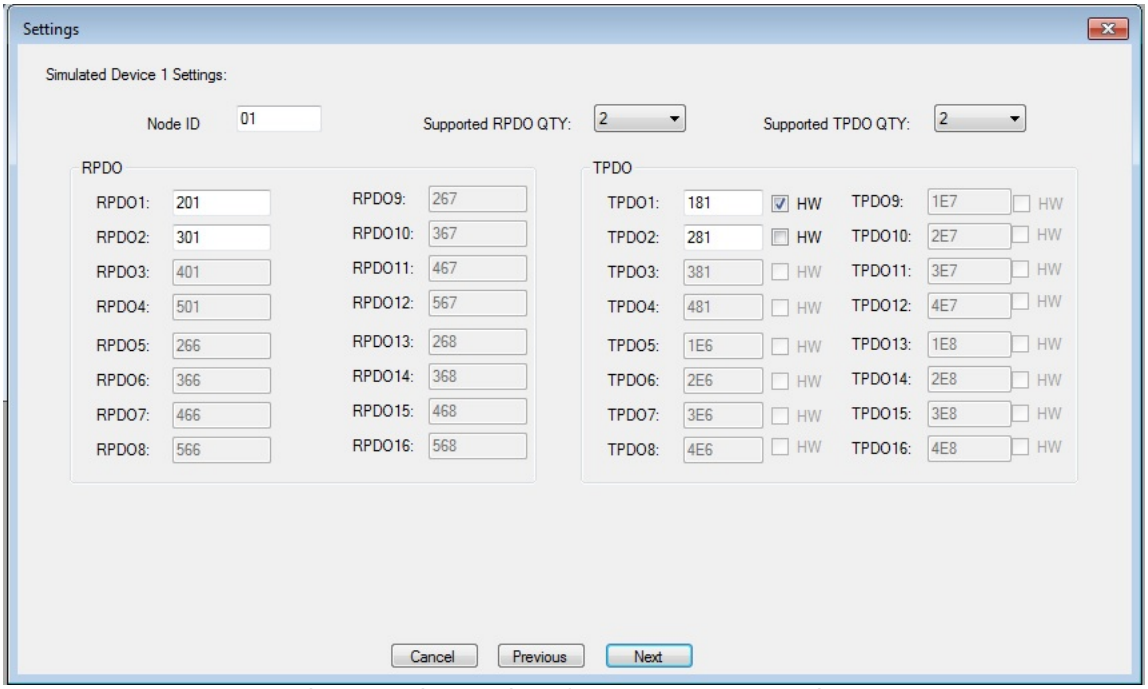


Fig.25 Device Settings for CANopen Protocol

These items are standard CANopen server items. All data are input in Hex. Max TPDO quantities are 16. Max RPDO quantities are 16

Your team worker may have set up all settings, you just use menu item : "File/Use Settings File ..." to use .dcfg suffix file. It will save your time. Or you want to save your settings for your co-worker use, just use menu item : "File/Save" or "Save as" to save to .dcfg file.

For CANopen protocol, you should use "Object Dictionary" menu to Edit your Dictionary, and you can save to .od suffix file. Of cause , you can use "File/Use OD File ..." to use external OD file. We will explain in the section of "One Example for CANopen".

2 In this stage, You will set up finite state machine and script for each device.

What is finite state machine?

"Finite State Machine" (or called "State Machine") is a mathematical model of computation. In any time, system always runs exact only one state of all finite states, the other states are not running.
The jump from one state to another state needs some conditions. When conditions are met, system will execute special action and jump to another state. Please see figure below. The initial State is "State 1". System executes code in State1.
When Condition is met, system executes code in Action and then enters "State 2" and executes code in "State 2"

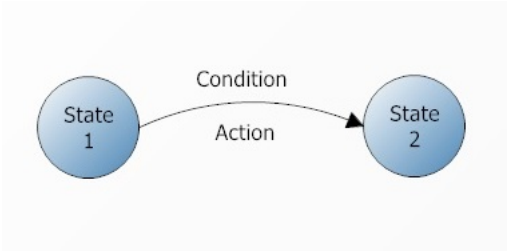


Fig.26 State Machine

We use simplified "Finite State Machine" which has no jump condition and action. All jumps are unconditional and no any action. Please see figure below, its function is the same as figure above.

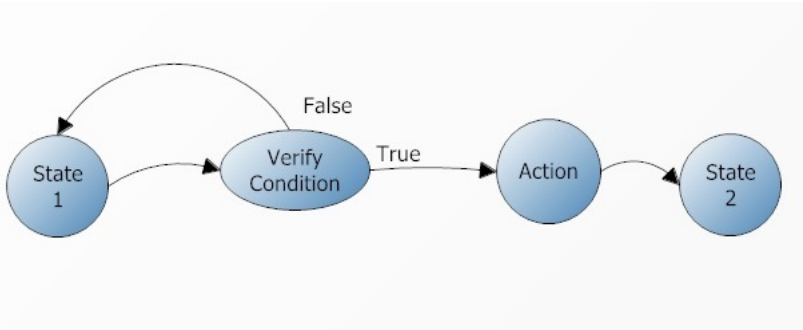


Fig.27 Simplified State Machine

We add a special state "Verify Condition". When Condition is true (code inside this state decides variable "Verify" value true), it will enter state "Action", other return to "State 1". And code inside

"State 1" completes, it will enter "Verify Condition" again unconditionally.
All jumps in the simplified "State machine" are all unconditional.

In general, the result of code running (variable or array value) is non-volatile. So we can use the following state machine to replace it.

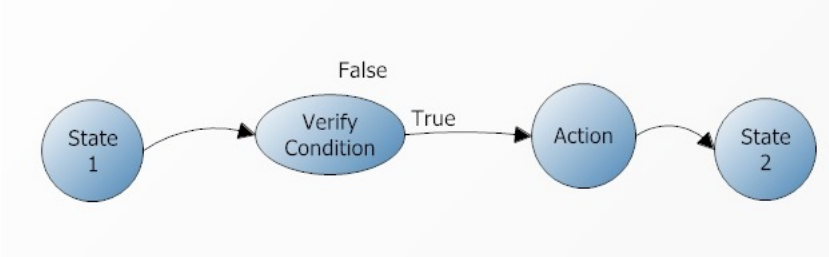


Fig.28 Simplified State Machine

Our State machine is this kind of simplified "Finite State Machine". In the left tool box, we provide 5 kinds of state: "Fault State", "Verify State", "Delay State", "Action State", and "Warn" State.

Let us explain these states:

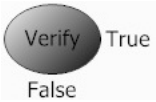
: Fault State. When any FaultInput(i) (i=0 to Fault Input Qty -1) is true, it will enter this state automatically (no need any connection line to enter). And state color becomes Red, and run VB Script code inside this state. When all FaultInput(i) (i=0 to Fault Input Qty -1) is false, it will exit automatically, Which state it will exit to is decided by your state connection line.

: Delay State. When enter this state, it will display count-down time in ms and State color becomes Green. When count-down to 0, it will exit. Although delay time unit is ms, the actual time resolution is 150ms because Windows OS is not real-time OS. And the actual delay time is not accurate due to OS, you can decrease delay time to match actual delay time you want.

: Action State. When enter this state, it will run VB script code inside it once, State color becomes Green. and then exits. State color becomes Grey again.

: Warn State. It is similar to Fault State. When any WarningInput(i) (i=0 to Warning Input Qty -1) is true, it will enter it automatically. And color becomes yellow, and run VB Script code inside this state. When all WarningInput(i) (i=0 to

Warning Input Qty -1) is false, it will exit automatically, Which state it will exit to is decided by your state connection line.



: Verify State. When enters this state, it will display count-down time (time out) in ms and State color becomes Green. When condition variable "Verify" in VB Script becomes true, it will exit automatically, Which state it will exit to is decided by your state connection line to "True point". If count-down to 0 and "Verify" is still false, it will exit automatically,Which state it will exit to is decided by your state connection line to "False point". When we set up time out=0, the "Verify" State will execute once to decide next state. Of cause, if "False" does not has any state connected, it will wait for true forever. "Verify State" can do extra action like "Action" State. But don't transmit any CAN bus packet. The left side of "Verify State" (the horizontal line of "True") is True although there is no label True. The Top side of "Verify State" (the vertical line of "False") is False although there is no label False.

Now we can set up state machine for simulated device. For new devices, you should know its principle, and divide it into many different states. VB Script for every state only needs a few of assignment statements. (If you only use one State which is "Action State", all functions are put into complex scripts, it can work, But your job becomes very complex and not reliable).

Click menu item: "File/New State Machine" will Open one or more empty state machine windows, Each state machine window denotes each device you simulate. See Screenshot below:

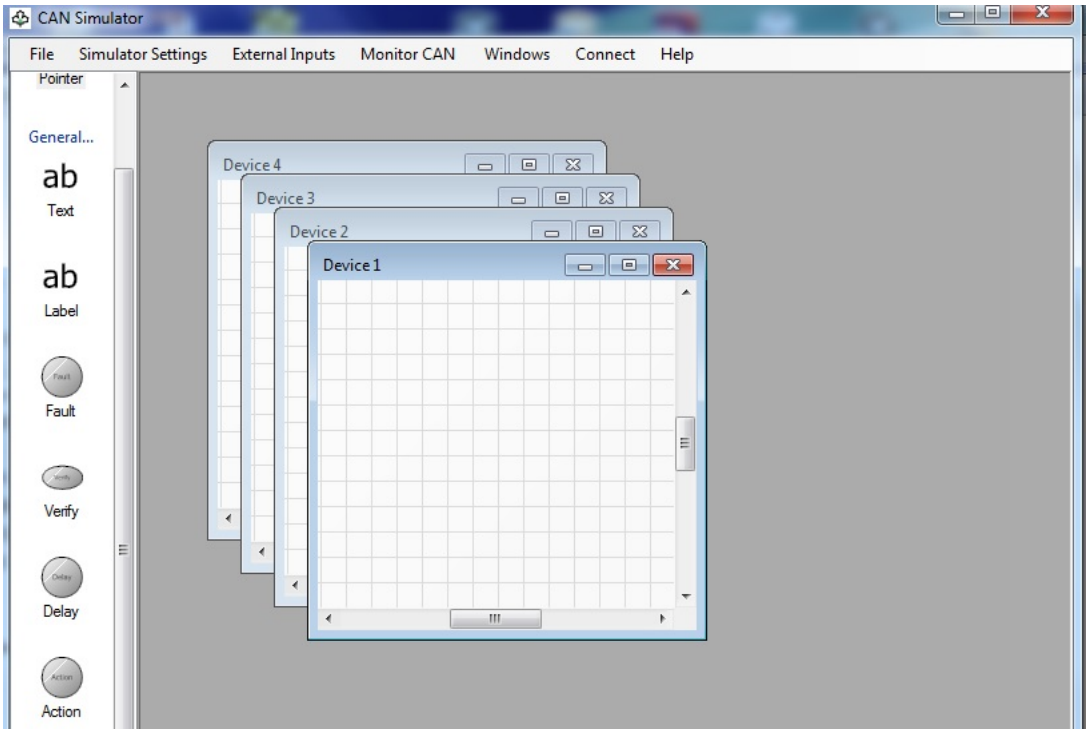


Fig.29 New State Machine

You can click Windows/Horizontal Title or others to organize your State machine window. Firstly we work in Device1 (Windows title is Device 1), Maximize this window, and click state icon in the State Tool box, and click on the client position of Device 1 window. This state will be put on Device 1 state machine. See result for "Action" State as example below:

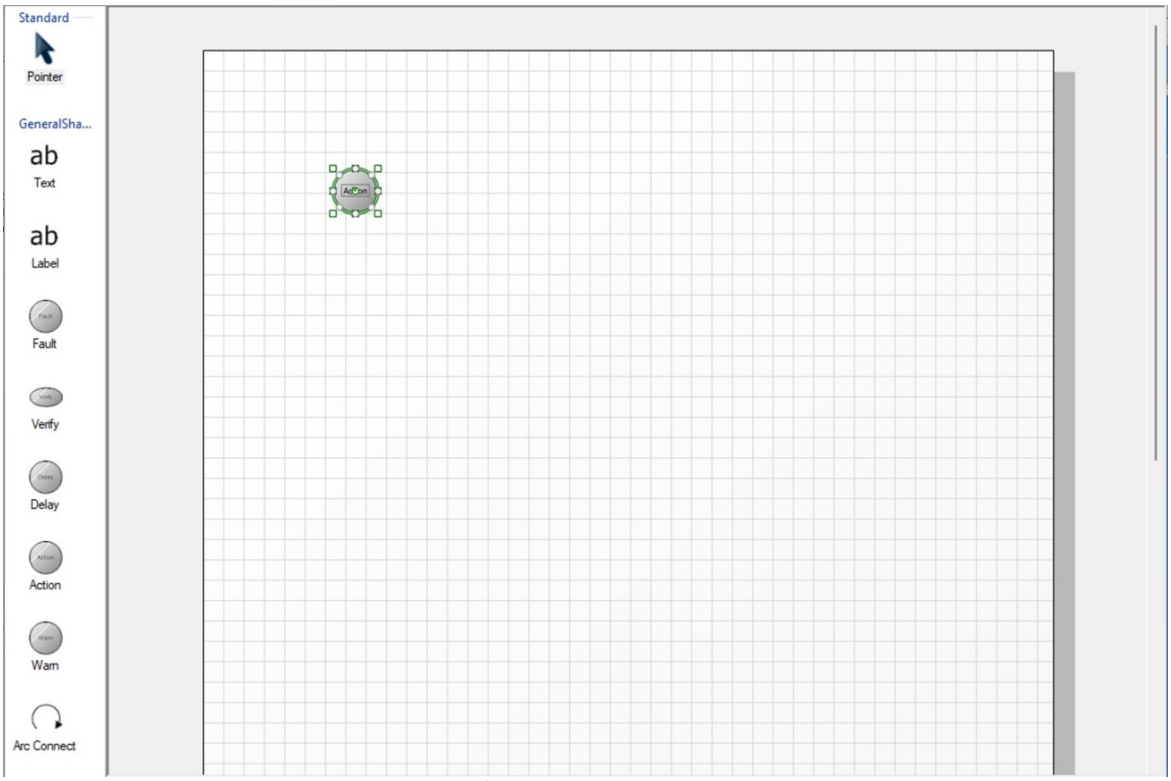


Fig.30 put one state

The 8 green squares can be used to adjust state picture size. You can click on the state name to change default name to any text instead of "Action". You can drag this state to any position. Single Click on this state will make border of state to become thick, which means it is initial state. If you don't want to change initial state, click menu item "File/ Initial State Lock" to lock initial state. We create 7 states by this way , see screenshot below:

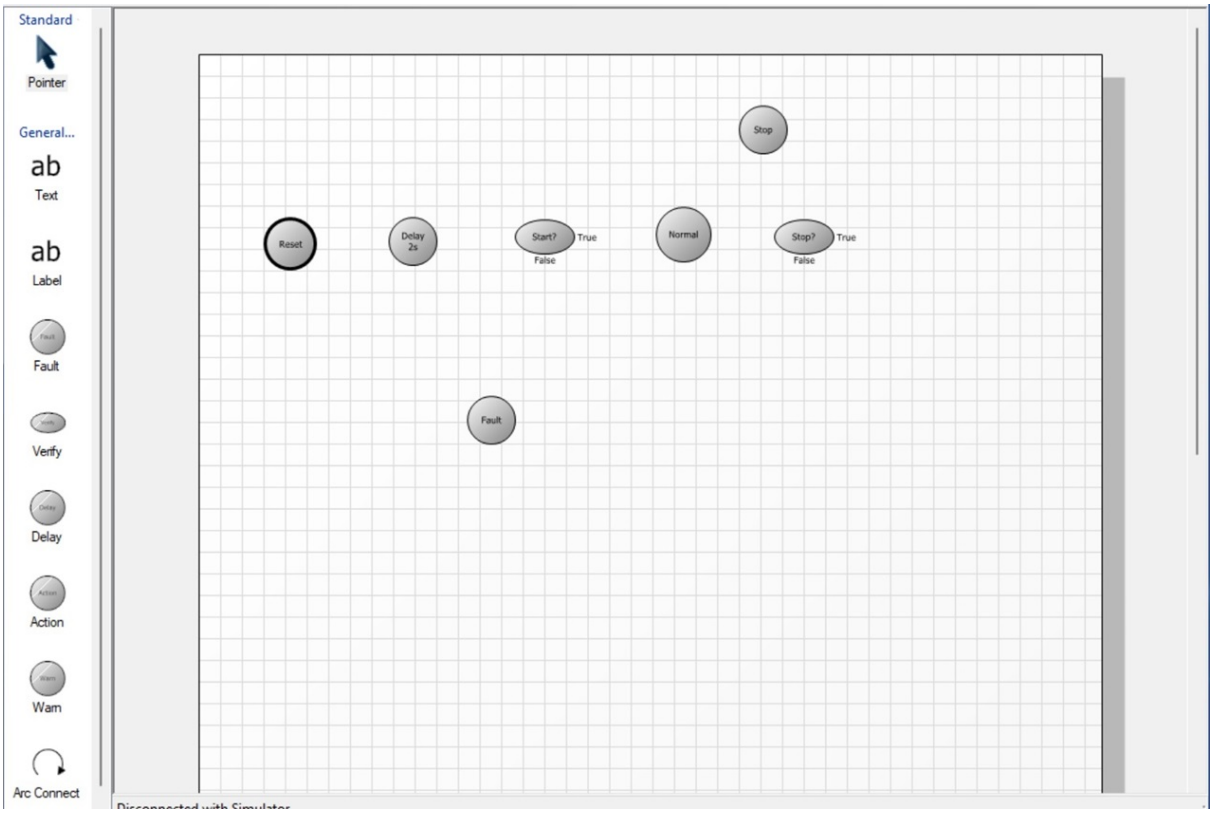


Fig.31 All States

The 7 states are "Reset" (Action), "Delay 2s" (Delay), "Start?" (Verify), "Normal" (Action), "Stop?" (Verify) , "Stop" (Action) and "Fault" State.

The basic function is that Power on device and then device transmits one special CAN Bus packet and delay 2 seconds, and then detect start command. Device will enter Normal run if start command receives. And device will detect stop command. Device will enter Stop state if stop command receives. And then detect start command again, ... No matter what state device is, device will enter fault state if fault occurs. Device will enter reset state if fault disappears.

This is typical device. Click "Arc connect" icon in state machine tool box. And then put cursor on one of small square of "Reset" state, Cursor will become a small hand,and then click on "small hand". Move cursor to one of small square of "Delay" state. Cursor becomes a small hand, click on "small hand". The connection between "Reset" and "Delay" will be set up. The connection maybe not good like below:

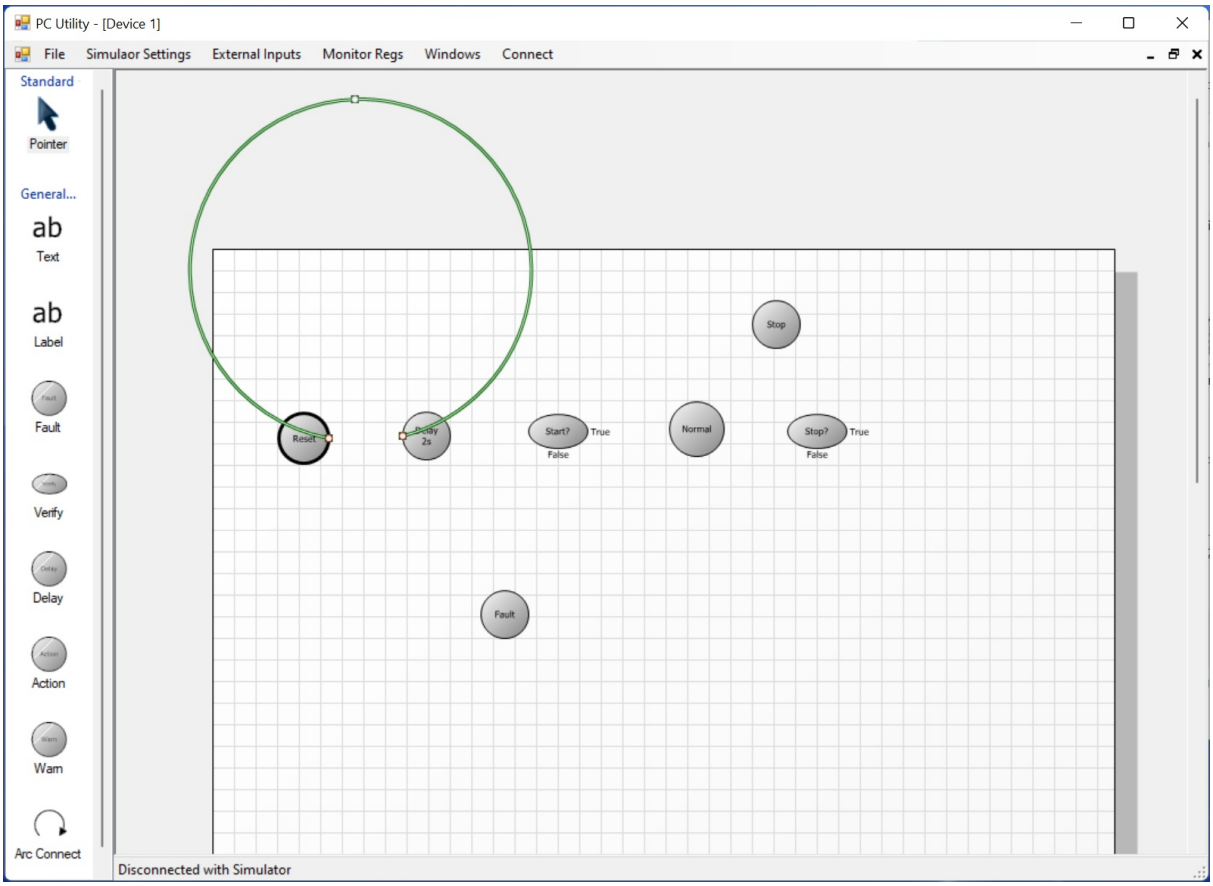


Fig.32 State Connection

Don't worry, just drag middle square of "Arc Connect" line to good position such as screenshot below:

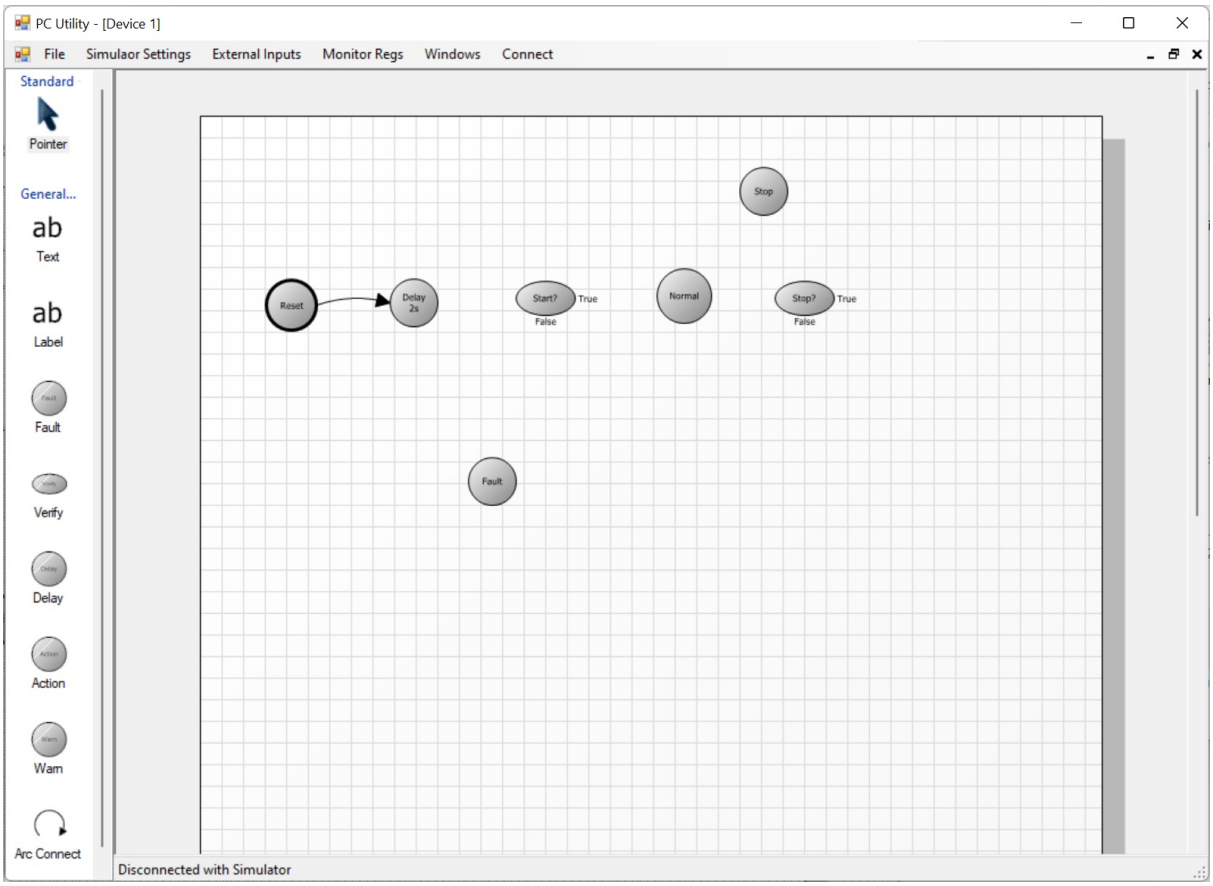


Fig.33 State Connection

We complete all states connections like below:

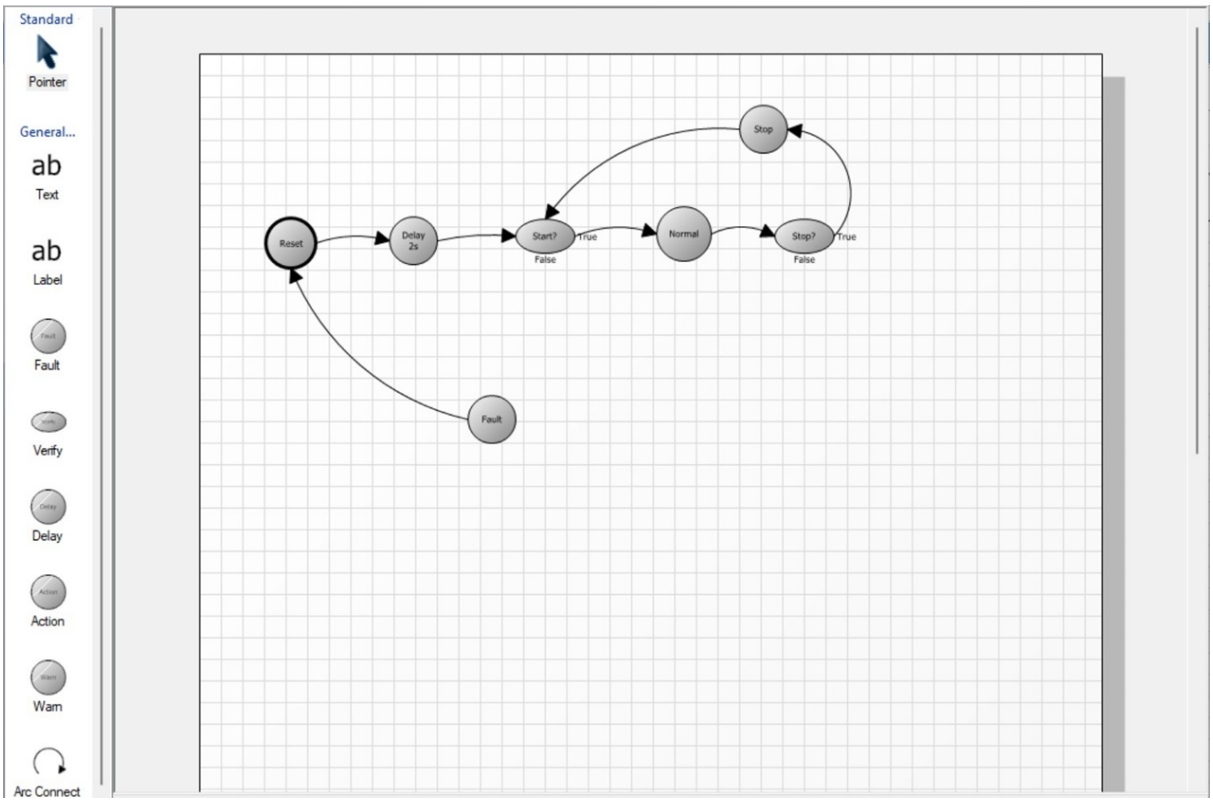


Fig.34 All States Connections

***Notes**: Sometimes, your connection is not good although visibly it is connected. You can verify whether connection is good by moving state. If state moves and its connection line moves with it automatically, it means connection is OK.*

You can add comment by "ab Text" icon, or "ab" Label" . What difference between "ab Text" icon, and "ab Label" ? "ab Text" is for entire state machine, it is not attached to any state or "Arc connect". However, "ab Label" is for one state or one "Arc Connect", it is attached to one state or "Arc Connect". When attached state or "Arc Connect" moves, "ab Label" moves with it automatically.

Now lets design scripts.

Script language is VB. Its Syntax is simple. It is almost the same as Microsoft VB.NET. However, it has some difference. In Dim statement, you cannot use Uint16 to replace UShort , cannot use Int16 to replace Short, cannot use Int32 to replace Integer, cannot use UInt32 to replace UInteger. Most of complex predefined class will be not supported. This VB script does not support "iif()" function. Any variable must be defined by "Dim or Static" statement before it is used.

Every Device supports the following global variables or array for CAN BUS Protocol:

- **FaultInput()**: 32 fault inputs, Type is Boolean (read/write), index is from 0 to 31 , which denotes the 1st to 32th Fault inputs. You can change its value by menu "External Inputs"/"Fault and digital inputs". Of cause, you can set/clear by your script.
- **WarningInput()**: 32 Warning inputs, Type is Boolean (read/write), index is from 0 to 31 , which denotes the 1st to 32th Warning inputs. You can change its value by menu "External Inputs"/"Warning inputs". Of cause, you can set/clear by your script.
- **AnalogInput()**: 32 analog inputs, Type is single (read only), index is from 0 to 31 , which denotes the 1st to 32th analog inputs. You can change its value by menu "External Inputs"/"Analog Inputs".
- **RawAnalogIn()()**: Raw data value of 32 analog inputs, Type is 32 bits of unsigned integer (read only). The first index is from 0 to 31 , which denotes the 1st to 32th analog inputs. The second index is from 0 to 3, which denotes the 1st to 4th byte of raw data (little Endian). Its value is decided by AnalogInput() and scale/offset in device configuration
- **DigitalInput()**: 10 Digital inputs, Type is Boolean (read only), index is from 0 to 9 , which denotes the 1st to 10th Digital inputs. You can change its value by menu "External Inputs"/"Fault and digital inputs".
- **Verify**: "Verify State" result, Type is Boolean (write only). This is only for "Verify state" script.
- **Debug()**: 32 Debug inputs/outputs, Type is 16 bits of unsigned integer (read/write), index is from 0 to 31 , which denotes the 1st to 32th Debug. It can be used in "Debug" or "exchange Data between devices". You can view their values by menu"Windows/Debug" .
- **DebugSng()**: 32 Debug inputs/outputs, Type is single precision float (read/write), index is from 0 to 31 , which denotes the 1st to 32th Debug. It can be used in "Debug" or "exchange Data between devices". You can view their values by menu"Windows/Debug" . Note, DebugSng() has nothing to do with Debug()

Every Device of CAN BUS Protocol supports the following global variables or array :

- **RxCANFrame()()** : CAN Bus received frame. Type is 32 bits of unsigned integer (read only). The first index is CAN bus received packet number from 0 to Qty-1. You can directly use CAN ID String to replace packet number (If there is prefix 0x , it is hex, otherwise it is decimal). Script compiler will translate it into packet number automatically. The 2nd index is used for distinguishing CAN BUS packet field. 0 is for CAN ID. Bit 31 denotes whether received this CAN bus packet (0 : received, 1: not received). For example, we do a judgment of whether received CAN ID="0x17F". The script will be

```
If (RxCANFrame("0x17F")(0) AND &H80000000))=0 Then
    ' CAN ID="0x17F has received,
End If
```

The 2nd index=1 denotes Data length of CAN bus Data packet. Bit 7 is flag for RTR (1: RTR, 0:Data). Here Data length of CAN bus Data packet is different from CAN BUS DLC, It can be more than 8 (sum of all data bytes quantities including Data1 with the same CAN ID). In our simulator, Max DLC can be 64. When DLC>8, Data1 for each single CAN frame has special meaning which is configured in device settings on the right side of received CAN ID. In general, Data1 is used as frame number. But it can be any special meaning as you require. Simulator didn't interpret any Data1 of single CAN frame, it just put all data bytes into Data fields.
The 2nd index: From 2 to 65, denotes Data bytes including Data1 of every single frame.

- **TxCANFrame()()** : CAN Bus Transmitted frame. Type is 32 bits of unsigned integer (write only). The first index is CAN bus transmitted packet number starting from 0. You can directly use CAN ID String to replace packet number (If there is prefix 0x , it is hex, otherwise it is decimal). Script compiler will create index number and CAN ID data for you automatically. Maximum of the first index is 29, which means Transmitted CAN bus ID QTY is 30. The 2nd index is used for distinguishing CAN BUS packet field. 0 is for CAN ID (You don't need to set its value when first index uses CAN ID String). The 2nd index=1 denotes Data length of CAN bus Data packet. Bit 7 is flag for RTR (1: RTR, 0:Data). Here Data length of CAN bus Data packet is different from CAN BUS DLC, It can be more than 8 (sum of all data bytes quantities except Data1 with the same CAN ID). In our simulator, Max DLC can be 64. When DLC>8, Data1 for each single CAN frame is frame number starting from 0. It is different from RxCANFrame()(). Simulator removes data1 of single packet and combines into real data into Data fields. The 2nd index: From 2 to 65, denotes Data bytes. If DLC>8, it is pure data field by removing data1 of single packet. .

  ***Notes**: You don't need any send out statement in your script. The reason is Simulator will transmit CAN bus frame when you have any assignment statement about one CAN ID packet inside your script.*

- **COMErr**: Device Communication Error (Input), Type is Boolean (read only).

Every Device of J1939 Protocol supports the following global array:

- **RxJ1939()()** : J1939 received Packets, Type is byte (read only). The first index is J1939 Received Number from 0 to 9.
  RxJ1939(0)() is "PGN=60908 Cannot claim address", it can be used RxJ1939("60908n")() or RxJ1939("0xEE00n")() to replace.
  RxJ1939(1)() is "PGN=60908 Can claim address", it can be used RxJ1939("60908")() or RxJ1939("0xEE00")() to replace.
  RxJ1939(2)() is "PGN=60418 TP.CM global address", it can be used RxJ1939("60416g")() or

RxJ1939("0xEC00g")() to replace.

RxJ1939(3)() is "PGN=60418 TP.CM Local address", it can be used RxJ1939("60416")() or RxJ1939("0xEC00")() to replace.

RxJ1939(4)() is "PGN=60160 TP.DT global address", it can be used RxJ1939("60160g")() or RxJ1939("0xEB00g")() to replace.

RxJ1939(5)() is "PGN=60160 TP.DT local address", it can be used RxJ1939("60160")() or RxJ1939("0xEB00")() to replace.

RxJ1939(6)() is "PGN=59904 global address", it can be used RxJ1939("59904g")() or RxJ1939("0xEA00g")() to replace.

RxJ1939(7)() is "PGN=59904 local address", it can be used RxJ1939("59904")() or RxJ1939("0xEA00")() to replace.

RxJ1939(8)() is "PGN=59392 Ack global address", it can be used RxJ1939("59392g")() or RxJ1939("0xE800g")() to replace.

RxJ1939(9)() is "PGN=59392 Ack local address", it can be used RxJ1939("59392")() or RxJ1939("0xE800")() to replace.

In general, we don't use RxJ1939()() above because PC built-in software implement J1939 Protocol.

The 2nd index is used for distinguishing J1939 frame field.

0: source address.

1: DLC, Bit 7 is received flag (0: received, 1: not received). Bit3 to Bit0 is data byte quantities received. The statement for judgment of received PGN59392 should be the following:

```
If (RxJ1939("59392")(1) And &H80)=0 Then
    ' Received PGN 59392 Local handle
End If
```

2: Data1

3: Data2

4: Data3

5: Data4

6: Data5

7: Data6

8: Data7

9: Data8

- **TxJ1939()** : J1939 transmitted Packets, Type is 16 bits of unsigned integer (write only), The first index is J1939 transmitted Number from 0 to 28, which means J1939 transmitted Max QTY =29. You can use PGN String to replace it.

  The 2nd index is used for distinguishing J1939 frame field.

0: PGN. If your first index used PGN String, you don't need to set this PGN, script compiler will create it for you automatically. PGN string can be prefix with 0x, which means Hex, otherwise decimal PGN. For example, TxJ1939("61696")() or TxJ1939("0xF100")() will create TxJ1939(0)(0)=61696

1: Priority and Reserved and page bit. Bit0=Page bit, Bit1=Reserved bit, Bit4 to Bit2 is priority. You don't need to set it by Script because you already configured it in device settings,

2: Destination address, 0 to 253 for individual address, 255 for global address. You don't need to set it by Script

3: DLC, its value is 1 to 8, You don't need to set it by Script

4: Data1

5: Data2

6: Data3

7: Data4

8: Data5

9: Data6

10: Data7

11: Data8

***Notes**: You don't need any send out statement in your script. The reason is Simulator will transmit J1939 PGN frame when you have any assignment statement about one PGN inside your script.*

Every Device of CANopen Server Protocol supports the following global array:

- **RxCANOpen()()** : CANopen received Packets, Type is byte (read only). The first index is CANOpen Received Packet Number from 0 to 19.

  RxCANOpen(0)() is "RSDO", it can be used RxCANOpen("RSDO")() to replace, and easily to remember.

  RxCANOpen(1)() is "TimeStamp", it can be used RxCANOpen("TimeStamp")() to replace, and easily to remember.

  RxCANOpen(2)() is "Sync", it can be used RxCANOpen("Sync")() to replace, and easily to remember.

  RxCANOpen(3)() is "NMT", it can be used RxCANOpen("NMT")() to replace, and easily to remember.

  RxCANOpen(4)() is "RPDO1", it can be used RxCANOpen("RPDO1")() to replace, and easily to remember.

  RxCANOpen(5)() is "RPDO2", it can be used RxCANOpen("RPDO2")() to replace, and easily to remember.

  RxCANOpen(6)() is "RPDO3",it can be used RxCANOpen("RPDO3")() to replace, and easily to

remember.

RxCANOpen(7)() is "RPDO4",it can be used RxCANOpen("RPDO4")() to replace, and easily to remember.

.....

RxCANOpen(19)() is "RPDO16",it can be used RxCANOpen("RPDO16")() to replace, and easily to remember.

In general, we don't use RxCANOpen("RSDO")(), RxCANOpen("Sync")(), and RxCANOpen ("NMT")() above because PC built-in software implement CANOpen Server Protocol.

However, we usually use RxCANOpen("RPDO1")() to RxCANOpen("RPDO16")() to do application-specific task.

The 2nd index is used for distinguishing CAB Bus frame field.

0: DLC, Bit 4 is RTR flag (0: Data, 1: RTR). Bit3 to Bit0 is data byte quantities received. Bit 7 is received flag (0: received, 1: not received). The statement for judgment of received RPDO2 should be the following:

```
If (RxCANOpen("RPDO2")(0) And &H80)=0 Then
    ' Received RPDO2 Local handle
End If
```

1: Data1

2: Data2

3: Data3

4: Data4

5: Data5

6: Data6

7: Data7

8: Data8

You don't need to remember these variable names, you just need press "Ctrl+ Space" to display variable name in Script Editor.

One special State machine script is for "Delay state". The script is only constant. For example, Script is " 2400" means delay is 2400ms.

Double click on "Reset" State, a window pop out. see picture below:

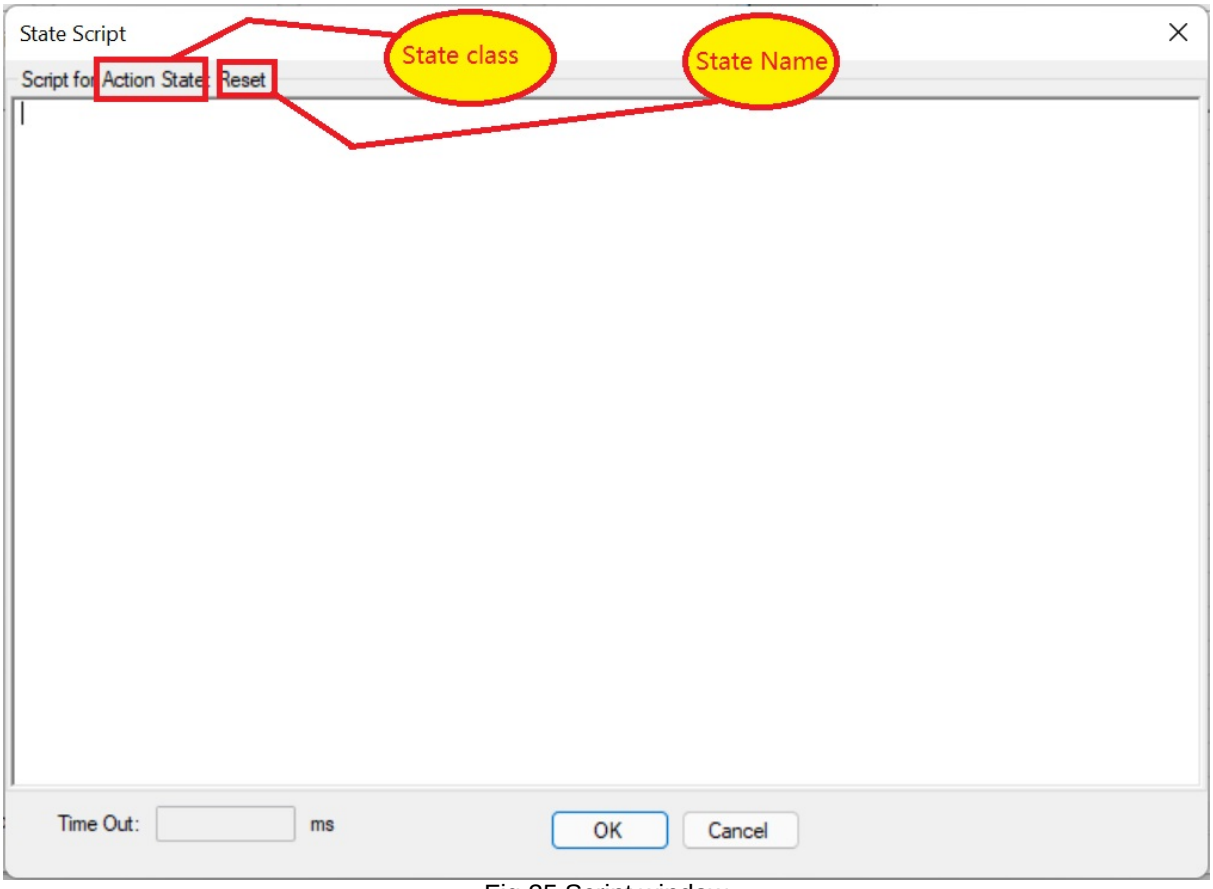


Fig.35 Script window

Our "Reset" State (Action) function is to transmit CAN packet: CAN ID=0x120, DLC=3, Data1=0x00, Data2=0x00 and Data3=0x01 . So we click on editor window, and press "Ctrl+ Space", variable hint occurs, click on "TxCANFrame" item,

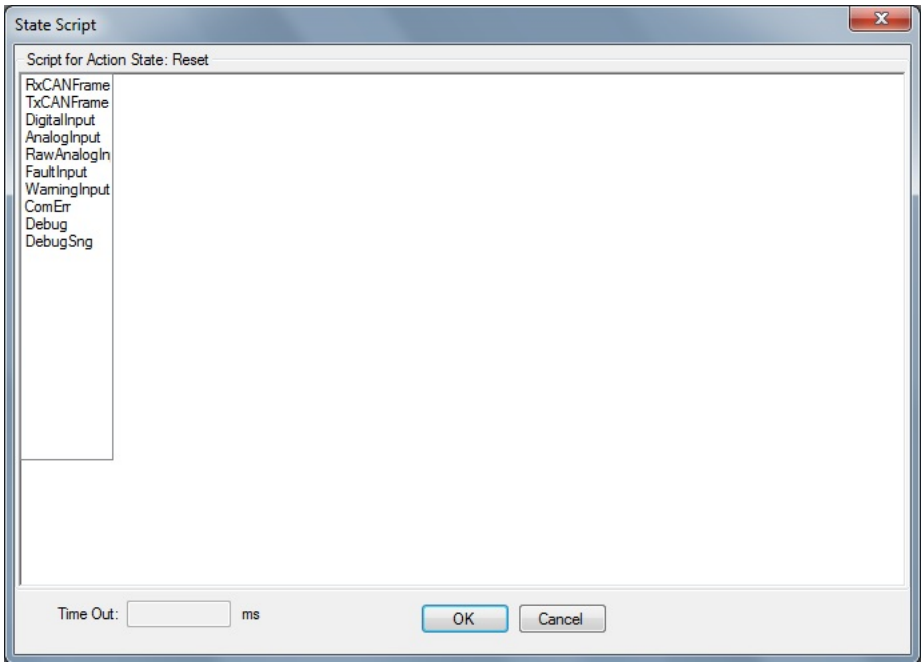


Fig.36 Script window

The script is shown below:

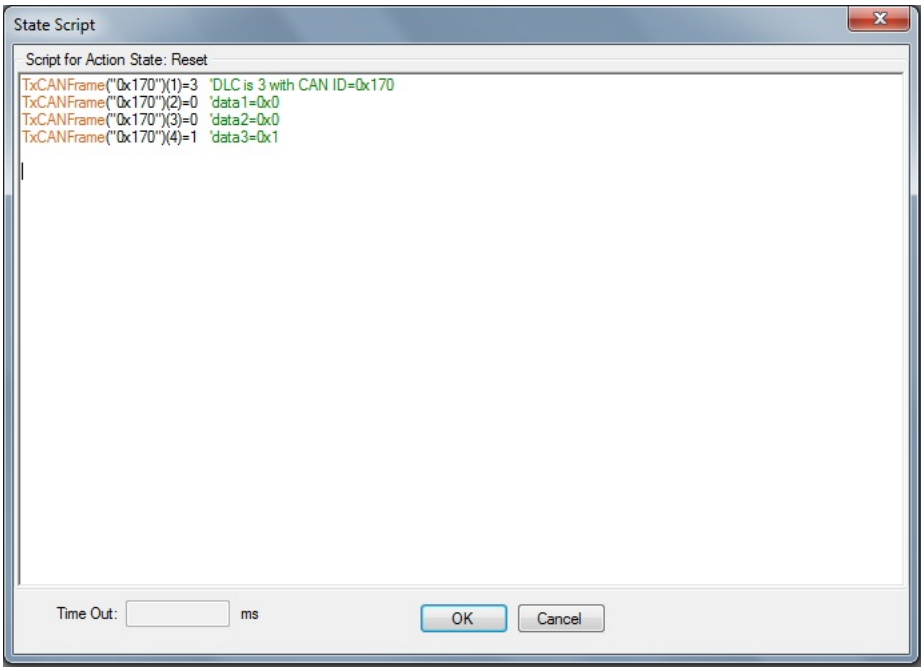


Fig.37 Script window

Click Button "OK" to accept script.

Notes: *1. There are 2 hints, one is for variables, the other is for statement keywords. These 2 hints alternates by "Ctrl+Space"*

2. *Script has syntax color. If it does not occur syntax color, please right click to click Syntax color menu item.*

Double click on "Delay 2s " (Delay) State. Pop out script window, we complete script such as window below:

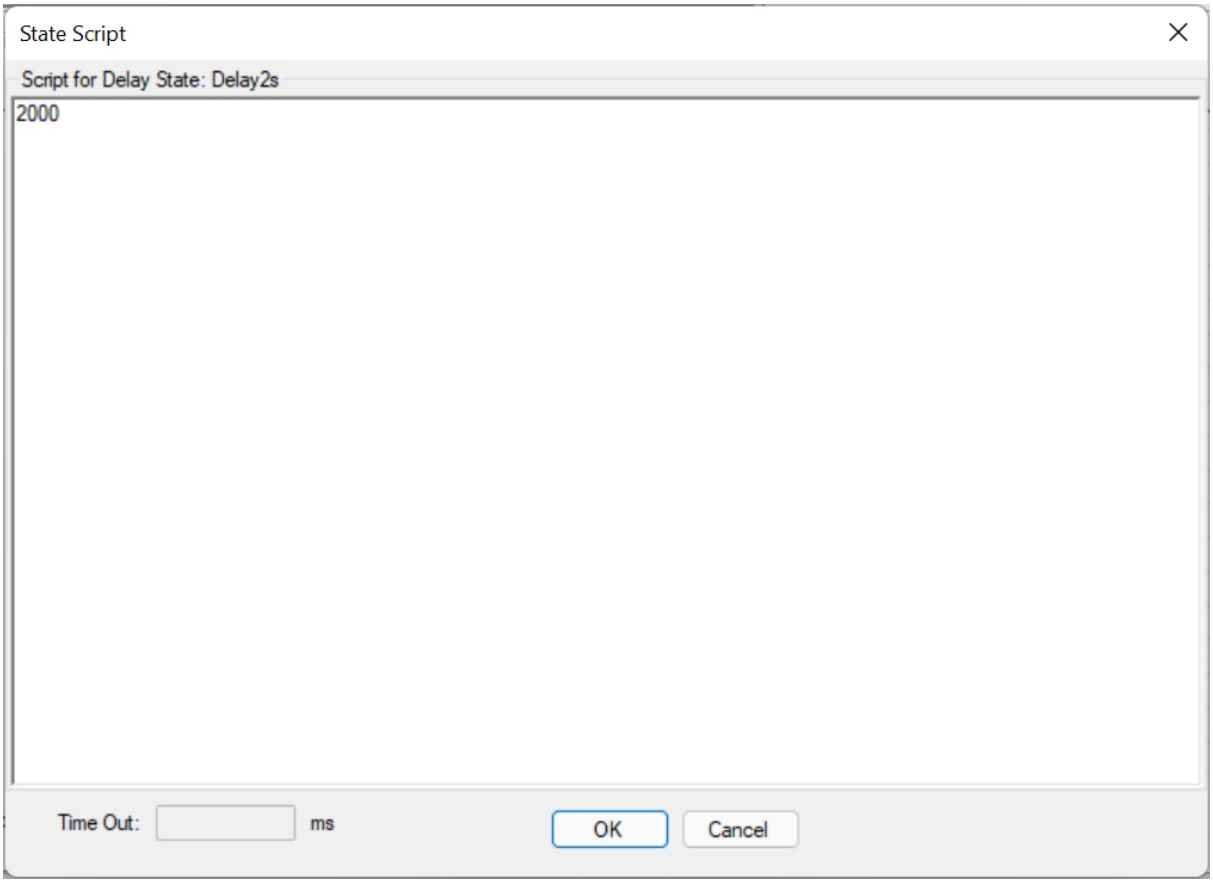


Fig.38 Script for "Delay State"

Notes: *If actual Delay is longer than 2 Seconds, please set this value lower than 2000. You should adjust this value according to actual delay time in order to get good delay time precision.*

Click Button "OK" to accept script.

Double click on "Start? " (Verify) State. Pop out script window, Our "Start?" state function is to see if CAN BUS ID=0X620 received and Data1=0x02. So Script content is shown below:

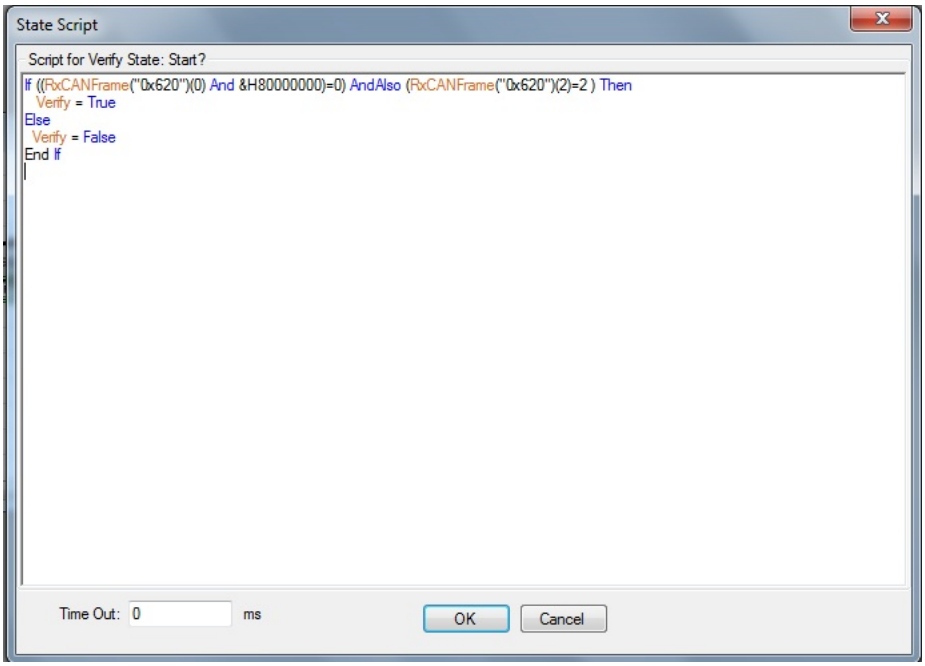


Fig.39 Script for "Verify State"

Time Out = 0 ms means that we wait for true forever. Click Button "OK" to accept script.

Double click on "Normal " (Action) State. Pop out script window, Write script as you need.
Double click on "Stop? " (Verify) State. Pop out script window, Write script as you need.
Double click on "Stop " (Action) State. Pop out script window, Write script as you need.
Double click on "Fault " State. Pop out script window, Write script as you need.

You can Copy/Paste Existing State by "Arrow" select icon and "Ctrl+C"/"Ctrl+V" . And you can use "Delete" key to remove some states or connections.
For example , we have 4 devices. and 4 devices are the same state machines. So we let Device 1 state machine window focused. Press "Ctrl+A" to select all elements in Device 1 state machine window, and press "Ctrl+ C" to copy all elements to clipboard. We let Device 2 state machine window focused. Press "Ctrl+ V" to paste all elements in device 1 to Device 2 State machine. We let Device 3 state machine window focused. Press "Ctrl+ V" to paste all elements in device 1 to Device 3 State machine. We let Device 4 state machine window focused. Press "Ctrl+ V" to paste all elements in device 1 to Device 4 State machine. This copy/paste includes Script.

To click menu item "Connect" will connect PC with simulator hardware, And for the first time, it will compile all scripts. See picture below:

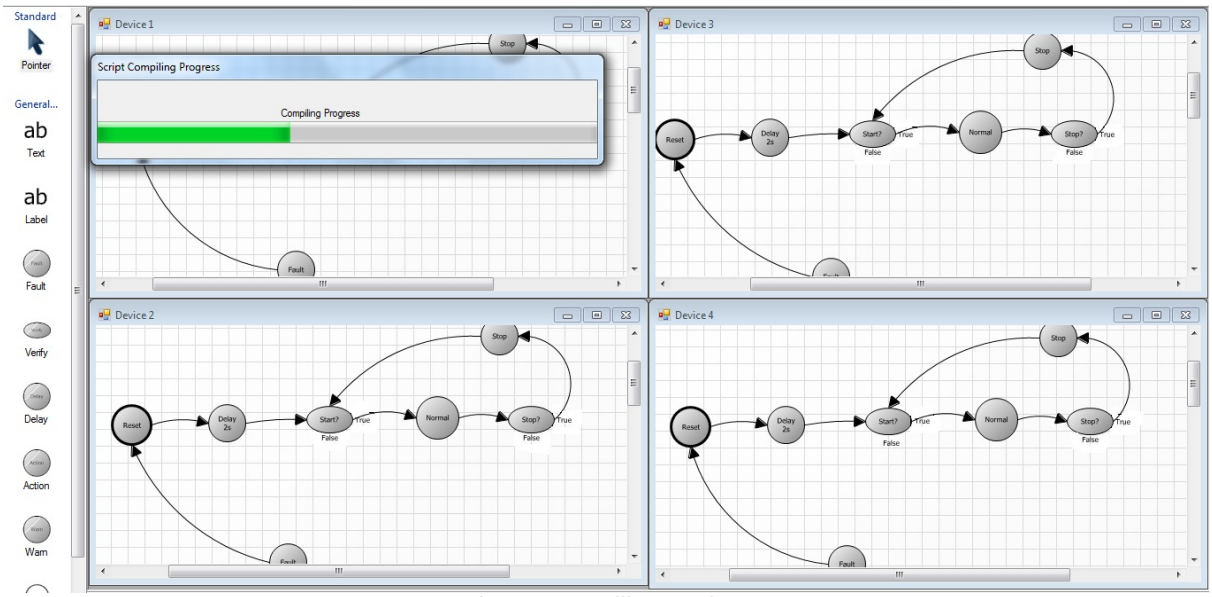


Fig.40 Compiling Scripts

After compiling successfully, it will simulate automatically. Please see picture below:

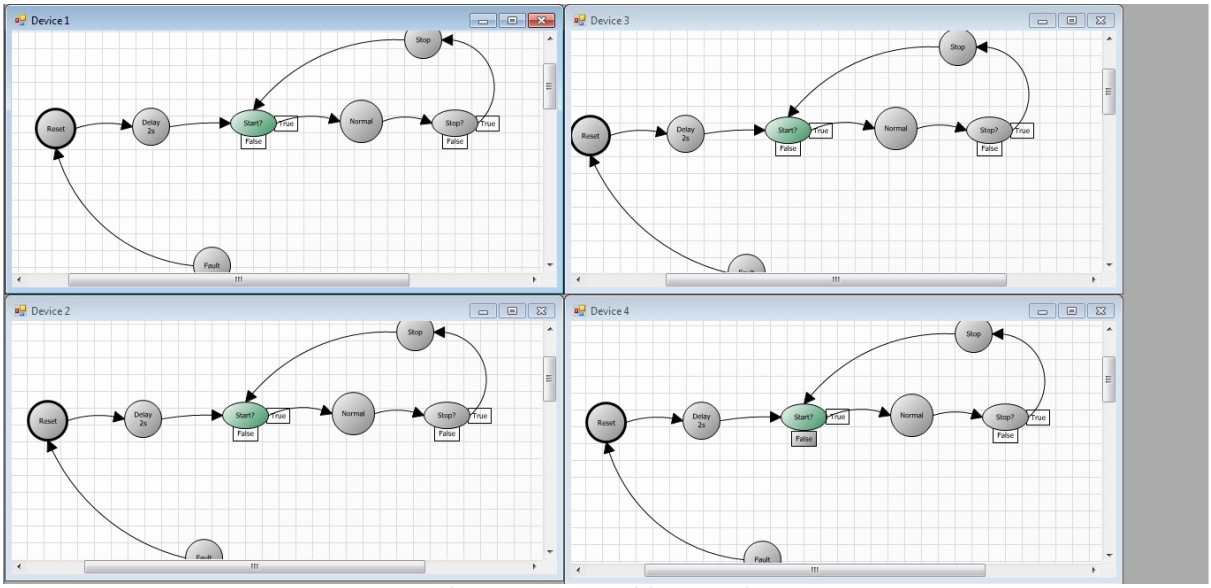


Fig.41 State machine running

The current state is in "Start?" which waits for outside start command. If you click menu item "External Inputs/Fault and Digital Inputs" to open fault inputs , click any fault to be checked in tab Device 1, Device 1 will enter fault State, See picture below:

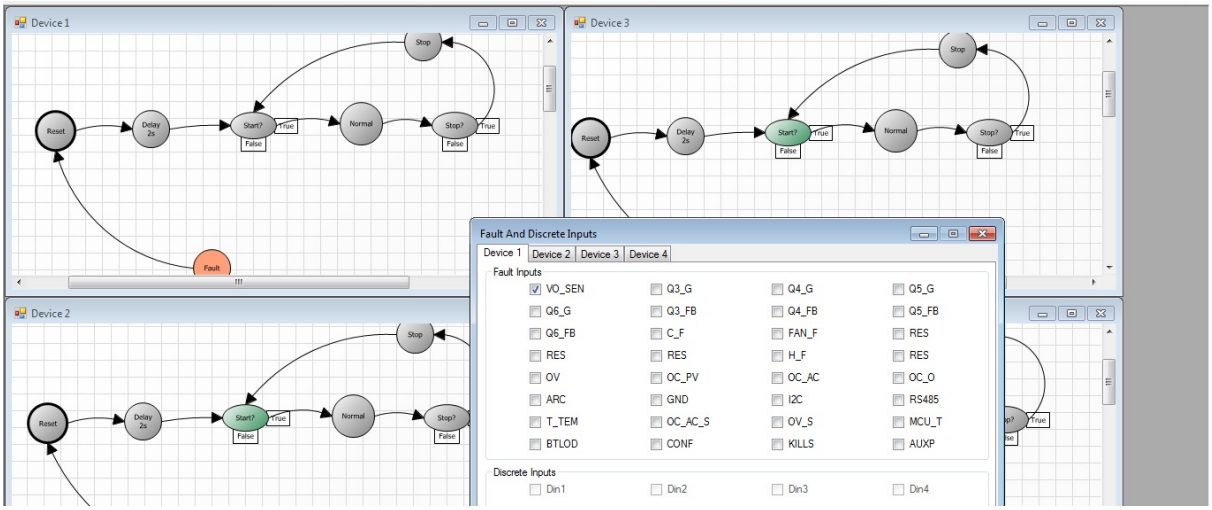


Fig.42 Fault occurs for device 1

At last, you can save your state machine for your team member use. Click item "Disconnect" to stop simulation, and then click menu item "File/Save State Machine or /Save State Machine As...." to save your state machine in *.nsprj file. Of cause, you can open state machine file to save time for drawing state machine and writing script.

For your Simulator, you should have 3 files if it is not CANopen protocol, otherwise, 4 files :

1. *. dcfg file for Device settings
2. *.nsprj file for state machine
3. *.scr file for script. This file name without the extension "scr" is the same as counterpart of *.nsprj. It automatically open or save when you open or save *.nsprj file
4. *.od file for Object Dictionary if we use CANopen protocol.

So you should copy 3 or 4 files above to your co-worker.

Notes: *Don't use "Arc connection" connecting One State to itself. If you really want to connect to itself, please use Delay State with delay time=0.*

5 One Example for CAN BUS

5.1 CAN Bus device description

We will demonstrate one device with 11 bits CAN ID and CAN baud rate =125Kbit/sec.

This CAN bus Device can receive 2 different CAN IDs, one is 0x17F and the other is 0x120

For CAN ID=0x17F which is RxWatchdog, DLC=1, watchdog byte length=1, Periods=600ms. RxWatchdog mode is 0. This RxWatchdog is used for "Synchronous function" too.
For CAN ID=0x120, Its DLC=8, and we will display it into debug(0) to debug(7).

This Device has 11 Analog inputs, call it CellV1 to CellV11. it is unsigned 16 bits integer, little endian, unit is V. and scale=0.0001V/bit, minimum=0V, Maximum=5V
This CAN bus Device can transmit one CAN ID, it is long frames, Data1 as frame number starting from 0. CAN ID is 11 bits of 0x140. Totally we have 22 bytes of pure data (not including frame number). This 22 bytes are 11 Analog Inputs raw data from CellV1 to CellV11.
This Device has two faults, call it Cell over Voltage fault (Simplified as OV) and Over Current fault (Simplified as OC).
When this device received RxWatchdog (Sync), it will transmit out 11 analog inputs. However, if any fault occurs, it will stop transmit any CAN Bus packets.

Any Cell Voltage (AnlogInput) >4.2, it will cause OV .

5.2 Setup Simulator

Let's setup this device simulator.

1 Please plug Simulator USB into your computer, and Run CANSimulator.exe, and then click Menu item: " Simulator Settings ". The following dialog occurs:

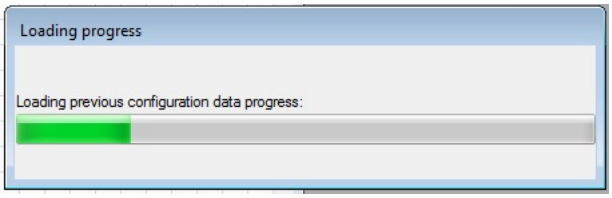


Fig.43 Load Settings

2 After loading settings, it will display below:

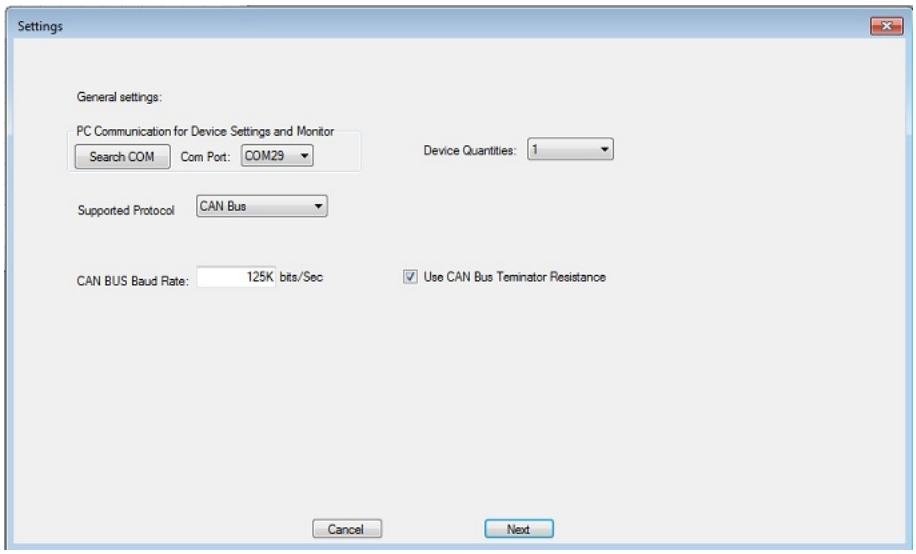


Fig.44 Protocol settings

We click "Search COM" button to select COM Port number from drop list, which is used for CAN Bus simulator. Select Device quantities to 1, and Supported protocol to "CAN Bus". CAN Bus baud rate is 125K. You can enable/disable 120 ohms terminator.
Click "Next" button to next step. Or "Cancel" button to exit settings (cancel all settings you made)

3 After above "Next" button click, the following window will display:

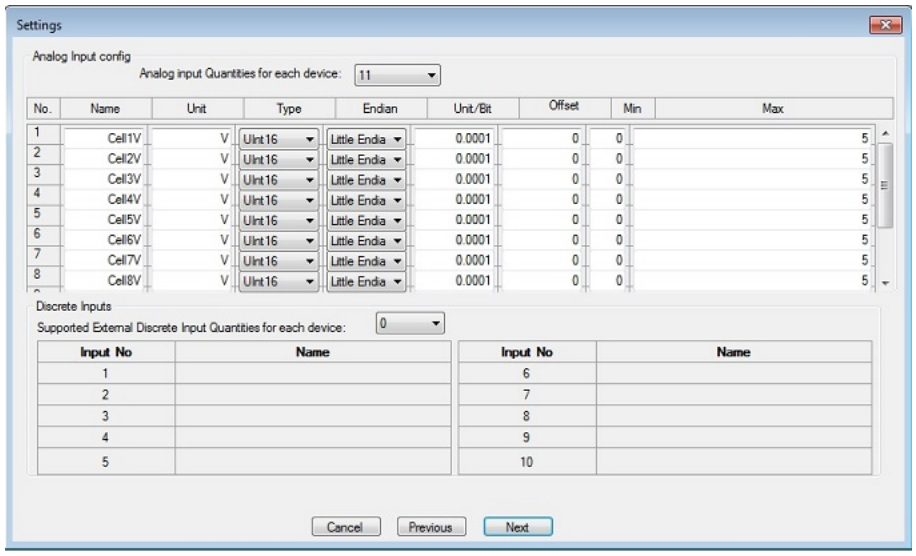


Fig.45 Analog/Digital Input settings

You select analog input quantities to 11, and set 11 analog inputs name from "Cell1V" to "Cell11V". Every analog input is 2 bytes's 16 bits unsigned integer (little endian). Offset=0V, Scale is 0.0001V/bit (That means value 10000 denotes 1V). No any digital inputs. Click "Next" to enter next step or "Previous" to return step 2 or "Cancel" button to exit settings (cancel all settings you made).

Notes: *If you simulate multiple devices, they will have the same analog inputs (or Digital inputs). Only values are different.*

4 In this step, you will set faults/warnings as display below:

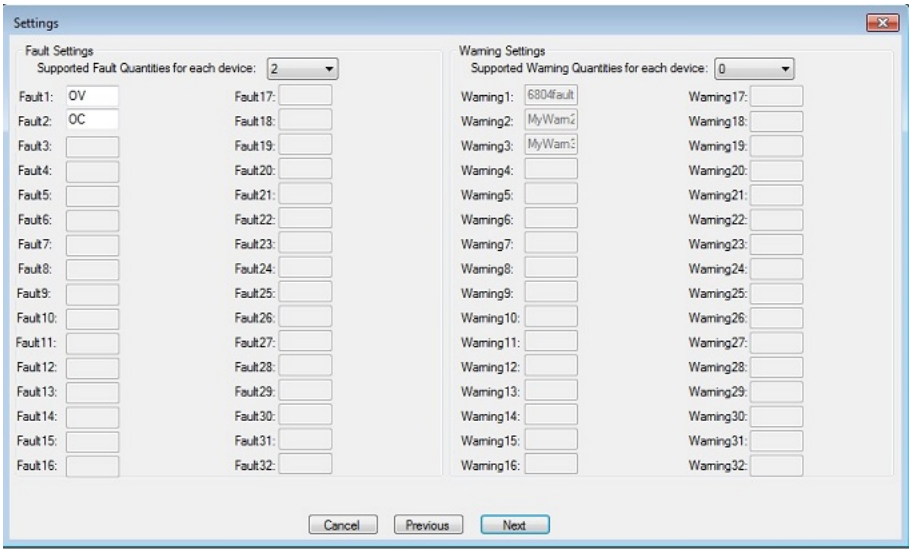


Fig. 46 Fault/Warning settings

We set 2 faults with names "OV" and "OC", no any warnings. Click "Next" to enter next step or "Previous" to return step 3 or "Cancel" button to exit settings (cancel all settings you made).

Notes*: If you simulate multiple devices, they will have the same faults (or warnings).*

5. In this step, we will set CAN ID to 11 bits, 5 filters (x as don't care) and Sync as below:

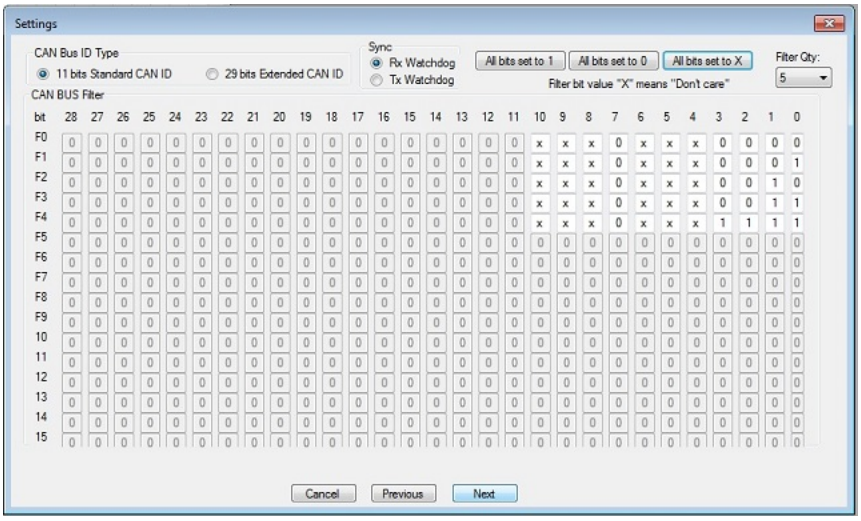


Fig 47 CAN bus type and filter settings

The 5 filters cover our 2 CAN bus ID: 0x17F and 0x120.
Of cause , we can choose 2 filters as below:

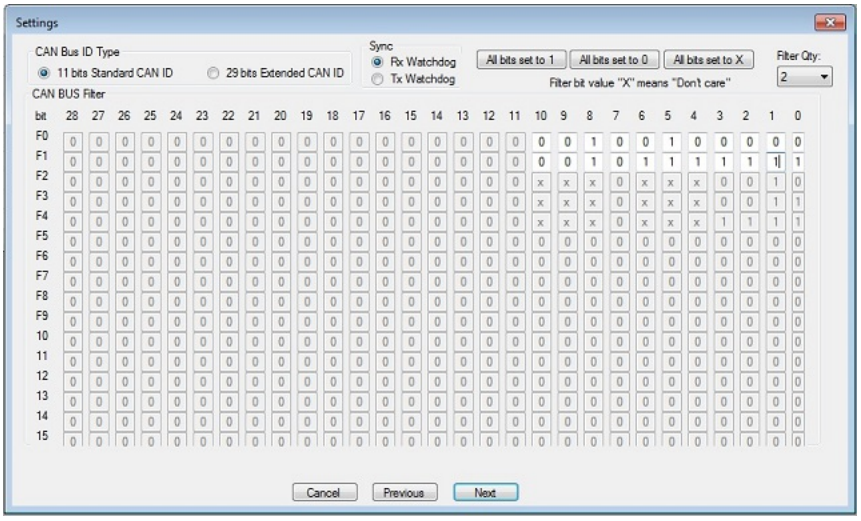


Fig 48 Another Filters settings

We used RxWachdog as Transmitting synchronous signal (The specific RxWatchdog CAN bus ID will be set in next step).

Click "Next" button to next step, or "Previous" to return step 4, or "Cancel" button to exit settings (cancel all settings you made)

6. In this step, we will set received CAN ID and RxWatchdog, TxWatchdog and Synchronous signal as below:

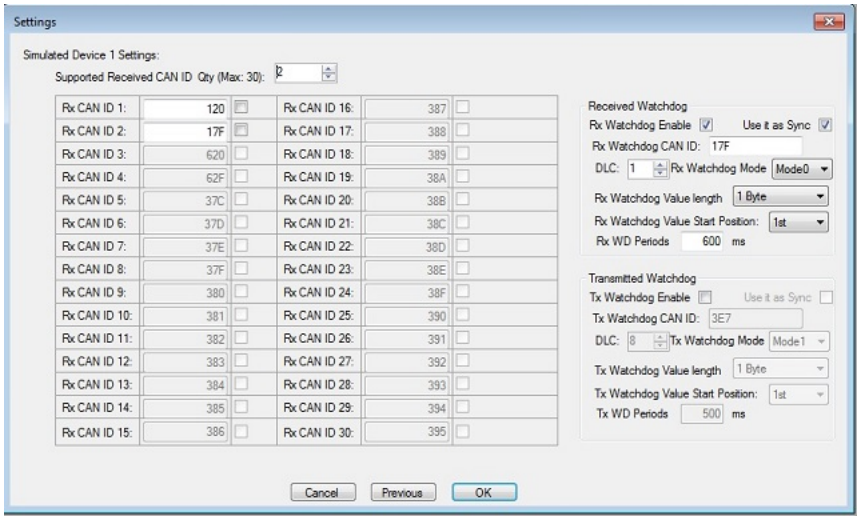


Fig. 49 Received CAN and watchdog settings.

We saw there is a check box beside each received CAN ID right side. If checked, it means the first data of CAN Packet as the same function as CAN ID (such as frame number), so it will display in different lines in "monitor CAN" window if the first data of CAN Packet is different even though they

have the same CAN ID. In this example, the first data of received CAN Packet is really data, so it is not checked.

Click "OK" button to finish all settings. Or "Previous" to return step 5, "Cancel" button to exit settings (cancel all settings you made)

***Notes**: If you have *. dcfg settings file, you can skip 1 to 6, just use File/Use settings File .....*

7. In this step, please click menu "File/New State Machine", and use left tools drag/drop to set up state machine as below:

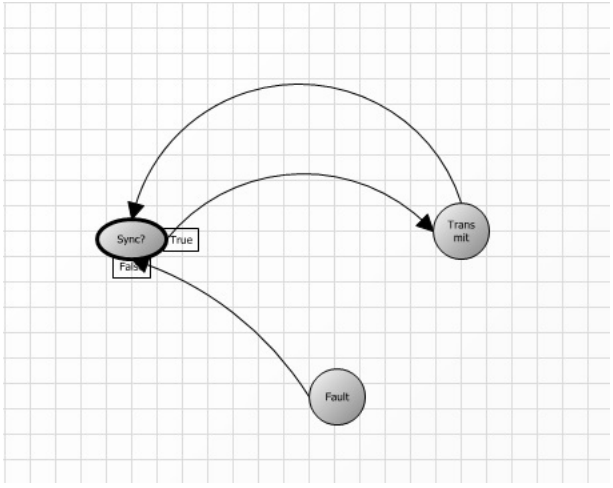


Fig. 50 State Machine

8. In this step, we will write every state's VB script. Double click on "Sync?" state, it will popup a editor window, and you type this script as below:

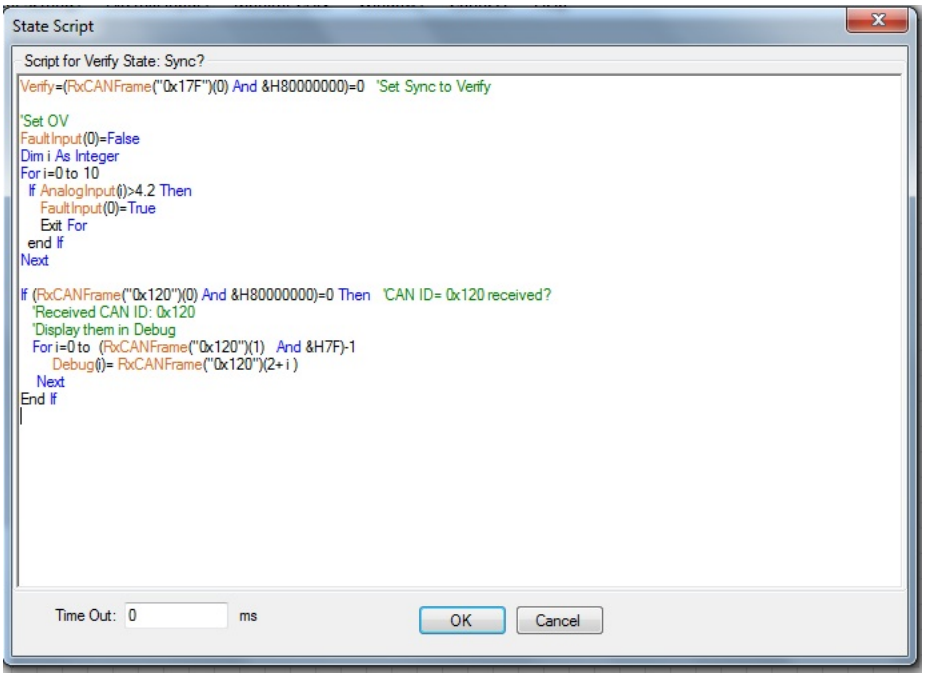


Fig. 51 Script for "Sync?" State

This script is for "Verify" state. The first statement will set "Verify" to true if CAN ID=0x17F is received. And middle part of code is for set the 1st fault. When any analog input is over 4.2V, it will set the 1st fault. The last part of code is to put Data with received CAN ID=0x120 into Debug registers.

Double click on "Transmit" state, it will popup a editor window, and you type this script as below:

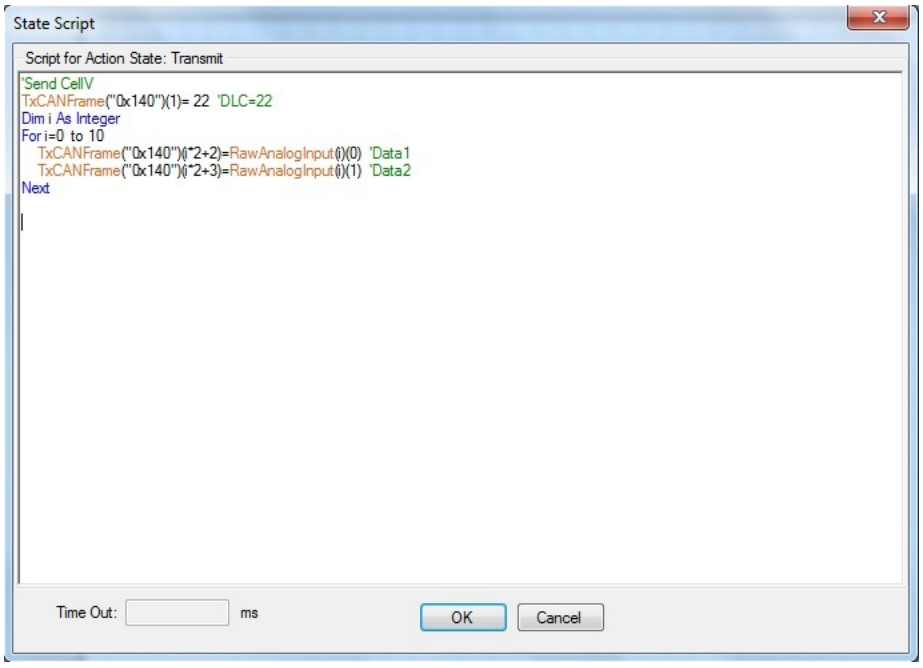


Fig. 52 Script for "Transmit" State

This script is for "Action" state. It will transmit CAN Packet with CAN ID=0x140. This CAN bus Packets are long frame, The 1st data is frame number starting from 0. The other data values are 11 Analog raw data (Little endian)

Double click on "Fault" state, it will popup a editor window, and you type this script as below:

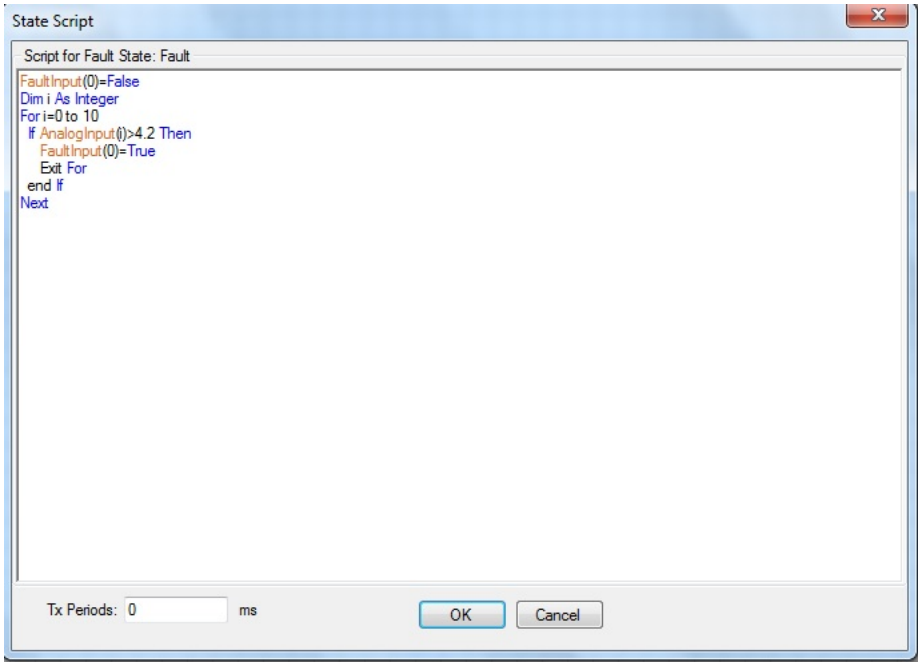


Fig. 53 Script for "Fault" State

This script is for "Fault" state. It will clear the 1st fault when all analog inputs<=4.2V, and it will set the 1st fault if any analog input>4.2V.

***Notes**: If you have *.nsprj *.scr state machine file, you can skip 7 to 8, just use File/Open State Machine .....*

9. Now we can click menu "Connect" to start simulator. The window below will be popped up:

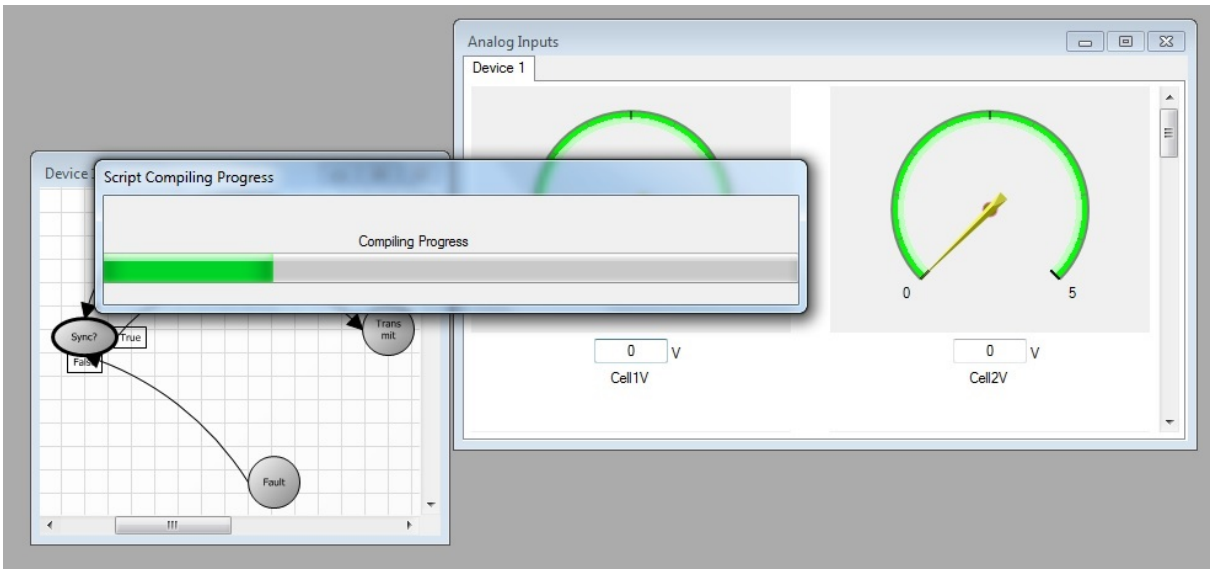


Fig. 54 Script compiling

It is compiling all state's scripts. If no any error for all scripts, State machine script will run. And we see "Sync?" is green color. It means device is waiting for Sync CAN Packet (CAN ID=0x17F). You can see monitor CAN Activity/ Analog inputs/Debug in real time. Please see Fig 55 below.

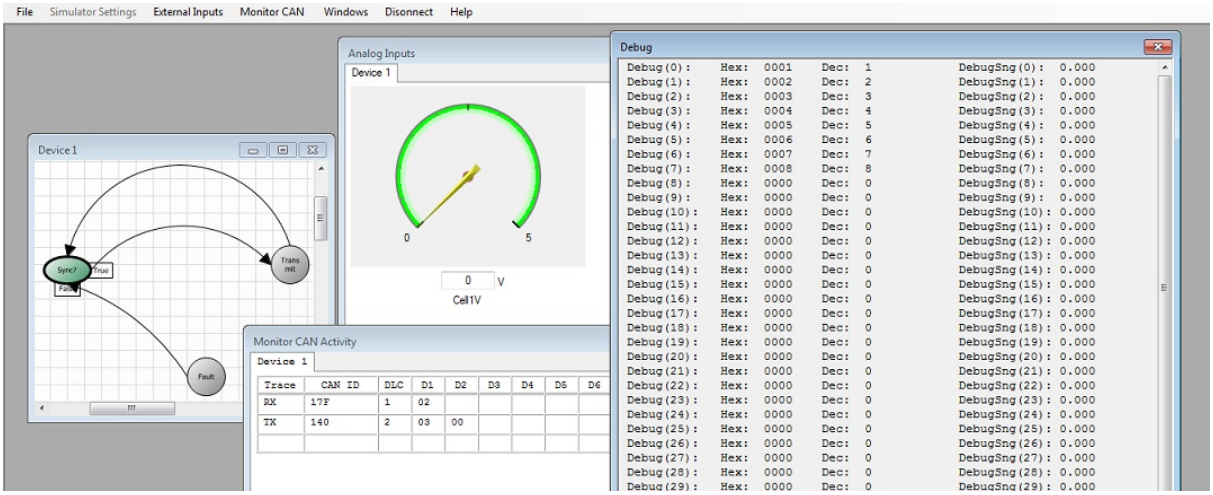


Fig. 55 State machine is running

In Fig. 55, we set Cell1V to 4,3V in "Analog Inputs" window, we can see State jump to "Fault" state, "Fault" state color becomes red. See Fig 56 below:

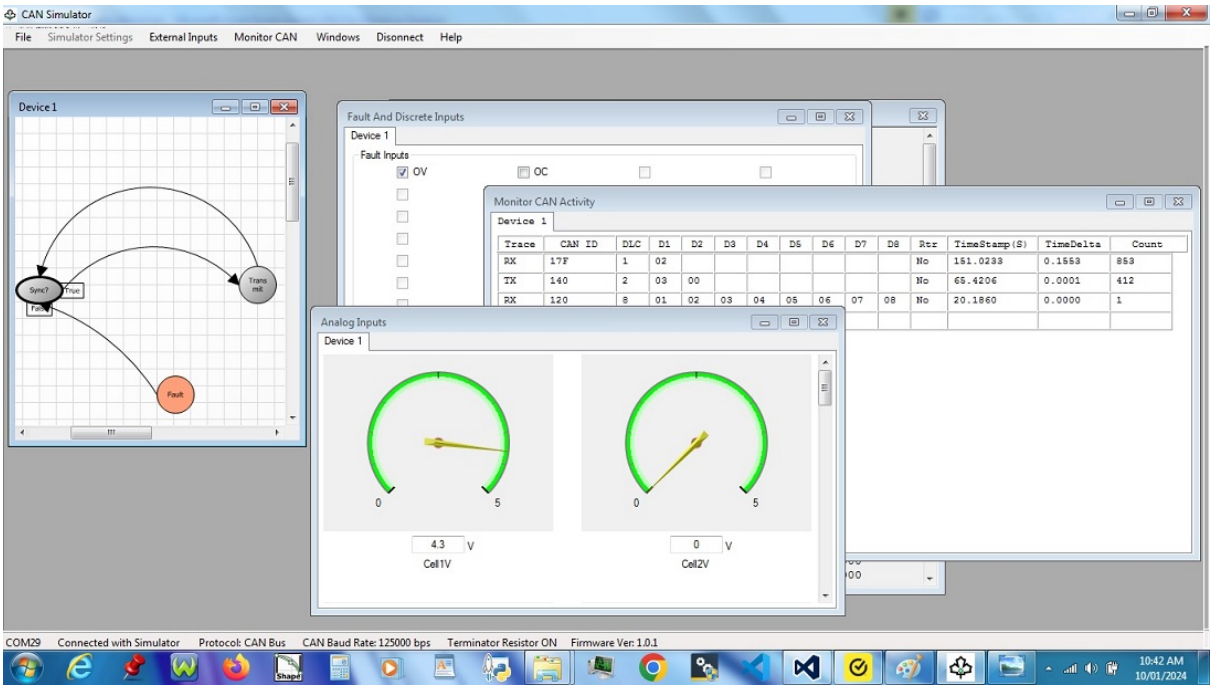


Fig. 56 Enter Fault State when Cell1V=4.3V

This example files are in folder: Top folder\CANbusSim\Example\CANBus

6 One Example for J1939

6.1 J1939 device description

We will demonstrate one example which implements function of our another hardware J1939 Simulator (Part Number: DFLSJ1939BV1)

CAN baud rate =250Kbit/sec. J1939 Address=0, Device Name=0x32FE084000201A0A VIN# is "DAFULA1ELECTR0N10"

Device has 29 Analog inputs below:

- The 1st Analog:
  Name: Latitude SPN=584, Unit: Degree, Type: unsigned int32, scale=10e-7 degree/bit, offset=210 degree
- The 2nd Analog:
  Name: Longitude SPN=585, Unit: Degree, Type: unsigned int32, scale=10e-7 degree/bit, offset=210 degree
- The 3rd Analog:
  Name:N_V_Speed SPN=517, Unit: Km/h, Type: unsigned int16, scale=0,00390625km/h/bit, offset=0 km/h
- The 4th Analog:

Name:Altitude SPN=580, Unit: m, Type: unsigned int16, scale=0.125m/bit, offset=2500m

- The 5th Analog:
  Name:Engine Coolant Temperature (CoolantTem) SPN=110, Unit: C, Type: unsigned int8, scale=1C/bit, offset=40C
- The 6th Analog:
  Name:Engine Fuel Temperature (FuelTem) SPN=174, Unit: C, Type: unsigned int8, scale=1C/bit, offset=40C
- The 7th Analog:
  Name:Engine Oil Temperature (OilTem) SPN=175, Unit: C, Type: unsigned int16, scale=0.03125C/bit, offset=273C
- The 8th Analog:
  Name:Barometric pressure (BaroPressu) SPN=108, Unit: KPa, Type: unsigned int8, scale=0.5KPa/bit, offset=0KPa
- The 9th Analog:
  Name:Ambient air temperature (AmbTem) SPN=171, Unit: C, Type: unsigned int16, scale=0.03125C/bit, offset=273C
- The 10th Analog:
  Name:Air inlet temperature (InletTem) SPN=172, Unit: C, Type: unsigned int8, scale=1C/bit, offset=40C
- The 11th Analog:
  Name:Engine trip fuel (TripFuel) SPN=182, Unit: Kg, Type: unsigned int32, scale=0.5Kg/bit, offset=0Kg
- The 12th Analog:
  Name:Engine total fuel used (TotalFuelUsed) SPN=250, Unit: Kg, Type: unsigned int32, scale=0.5Kg/bit, offset=0Kg
- The 13th Analog:
  Name:Actual engine - percent torque (ActualTorque) SPN=513, Unit: %, Type: unsigned int8, scale=1%/bit, offset=125%
- The 14th Analog:
  Name:Engine speed SPN=190, Unit: rpm, Type: unsigned int16, scale=0.125rpm/bit, offset=0 rpm
- The 15th Analog:
  Name:Accelerator Pedal Position 1 (AccePedalPos) SPN=91, Unit: %, Type: unsigned int8, scale=0.4%/bit, offset=0%
- The 16th Analog:
  Name:Engine Percent Load at Current Speed (Load@Speed) SPN=92, Unit: %, Type: unsigned int8, scale=1/bit, offset=0%
- The 17th Analog:

Name:Boost pressure SPN=102, Unit: KPa, Type: unsigned int8, scale=2KPa/bit, offset=0KPa

- The 18th Analog:
  Name:Tachograph vehicle speed (TachVehSpeed) SPN=1624, Unit: Km/h, Type: unsigned int16, scale=0,00390625km/h/bit, offset=0Km/h
- The 19th Analog:
  Name:Wheel-Based Vehicle Speed (WheelVehSpeed) SPN=84, Unit: Km/h, Type: unsigned int16, scale=0,00390625km/h/bit, offset=0Km/h
- The 20th Analog:
  Name:Electrical potential (ElecVoltage) SPN=168, Unit: V, Type: unsigned int16, scale=0.05V/bit, offset=0V
- The 21st Analog:
  Name:Fuel rate (FuelRate) SPN=183, Unit: L/h, Type: unsigned int16, scale=0.05L/h/bit, offset=0V
- The 22nd Analog:
  Name:Fuel delivery pressure (FuelPressure) SPN=94, Unit: KPa, Type: unsigned int8, scale=4KPa/bit, offset=0Kpa
- The 23rd Analog:
  Name:Engine oil pressure (OilPressure) SPN=100, Unit: KPa, Type: unsigned int8, scale=4KPa/bit, offset=0Kpa
- The 24th Analog:
  Name:Coolant pressure (CoolantPressure) SPN=109, Unit: KPa, Type: unsigned int8, scale=2KPa/bit, offset=0Kpa
- The 25th Analog:
  Name:Total engine hours (TotEngineHours) SPN=247, Unit: h, Type: unsigned int32, scale=0.05h/bit, offset=0h
- The 26th Analog:
  Name:Total engine revolutions (TotEngineRevol) SPN=249, Unit: r, Type: unsigned int32, scale=1000r/bit, offset=0r
- The 27th Analog:
  Name:Rated engine speed (RatedEngineSpeed) SPN=189, Unit: rpm, Type: unsigned int16, scale=0.125rpm/bit, offset=0 rpm
- The 28th Analog:
  Name:Trip Distance (TripDistance) SPN=244, Unit: Km, Type: unsigned int32, scale=0.125Km/bit, offset=0 Km
- The 29th Analog:
  Name:Total Vehicle Distance (TotDistance) SPN=245, Unit: Km, Type: unsigned int32, scale=0.125Km/bit, offset=0 Km

Device supports 3 Faults,

The first fault:

SPN 1208 is generated

SPN 1208 = 0x4B8 = 000 00000100 10111000 (19 bits)

FMI 3 = 3 = 00011 (5 bits)

OC 10 = 0xA = 0001010 (7 bits)

DTC=0x0097038A when we use DTC format version 1

DTC=0x9700038A when we use DTC format version 2

DTC=0xB804038A when we use DTC format version 3

DTC=0xB804030A when we use DTC format version 4

The second fault:

SPN 656 is generated

SPN 656 =0x290=000 0000 0010 1001 0000 (19 bits)

FMI 3 = 3 = 00011 (5 bits)

OC 2 = 0x2 = 0000010 (7 bits)

DTC=0x00520382 when we use DTC format version 1

DTC=0x52000382 when we use DTC format version 2

DTC=0x90020382 when we use DTC format version 3

DTC=0x90020302 when we use DTC format version 4

The third fault:

SPN 108 is generated

SPN 108 =0x6c=000 0000 0000 0110 1100 (19 bits)

FMI 11 = 0x0b = 01011 (5 bits)

OC 5 = 0x5 = 0000101 (7 bits)

DTC=0x000D8B85 when we use DTC format version 1

DTC=0x0D008B85 when we use DTC format version 2

DTC=0x6C000B85 when we use DTC format version 3

DTC=0x6C000B05 when we use DTC format version 4

Device supports 16 PGNs in addition to standard PGN59904/PGN60928/DM1/TP.CM/TP.DT/VIN/ DM11/PGN59392

We list these 16 PGNs below:

No.1

PGN 65267 Vehicle Position

Transmission Repetition Rate : 5 s

Data Length : 8

Reserve Bit : 0

Data Page : 0

PDU Format : 254

PDU Specific : 243

Default Priority : 6

Parameter Group Number : 65267 (0xFEF3)

Start Position	Length	Parameter Name	SPN
1-4	4 byte	Latitude	584
5-8	4 bytes	Longitude	585

With ECU simulated parameter value

Latitude: -210°(south) to 211.108122°(north) 10^{-7} °/bit

Longitude: -210°(west) to 211.108122°(east) 10^{-7} °/bit

No.2

PGN 65256 VEHICLE DIRECTION/SPEED

Transmission Repetition Rate :on request

Data Length : 8

Reserve Bit : 0

Data Page : 0

PDU Format : 254

PDU Specific : 232

Default Priority : 6

Parameter Group Number : 65256 (0xFEE8)

Start Position	Length	Parameter Name	SPN
1-2	2 bytes	Compass bearing	165
3-4	2 bytes	Navigation-based vehicle speed	517
5-6	2 bytes	Pitch	583
7-8	2 bytes	Altitude	580

With ECU simulated parameter value

Navigation-based vehicle speed: 0 to 250.996km/h 1/256 km/h/bit

Altitude: -2500m to 5531.875m 0.125m/bit

Another position are all 0xFF (not yet implemented)

No. 3

PGN 65262 Engine Temperature 1

Transmission Repetition Rate : 1 s

Data Length : 8

Reserve Bit : 0

Data Page : 0

PDU Format : 254

PDU Specific : 238

Default Priority : 6

Parameter Group Number : 65262 (0xFEEE)

Start Position	Length	Parameter Name	SPN
1	1 byte	Engine Coolant Temperature	110
2	1 bytes	Engine Fuel Temperature 1	174
3-4	2 bytes	Engine Oil Temperature 1	175
5-6	2 bytes	Engine Turbocharger Oil Temperature	176
7	1 bytes	Engine Intercooler Temperature	52
8	1 bytes	Engine Intercooler Thermostat Opening	1134

With ECU simulated parameter value

Engine Coolant Temperature: -40 to 210°C 1°C/bit

Engine Fuel Temperature 1: -40 to 210°C 1°C/bit .

Engine Oil Temperature 1: -273 to 1735°C 0.03125°C/bit

Another position are all 0xFF (not yet implemented)

No.4

PGN 65269 AMBIENT CONDITIONS

Transmission Repetition Rate : 1 s

Data Length : 8

Reserve Bit : 0

Data Page : 0

PDU Format : 254

PDU Specific : 245

Default Priority : 6

Parameter Group Number : 65269 (0xFEF5)

Start Position	Length	Parameter Name	SPN
1	1 byte	Barometric pressure	108
2-3	2 bytes	Cab interior temperature	170
4-5	2 bytes	Ambient air temperature	171
6	1 bytes	Air inlet temperature	172
7-8	2 bytes	Road surface temperature	79

With ECU simulated parameter value

Barometric pressure:0 to 125 kPa (0 to 18.1psi) 0.5kPa/bit

Ambient air temperature: -273 to 1735°C 0.03125°C/bit.

Air inlet temperature: -40 to 210°C 1°C/bit.

Another position are all 0xFF (not yet implemented)

No.5

PGN 65257 FUEL CONSUMPTION

Transmission Repetition Rate : 1 s

Data Length : 8

Reserve Bit : 0

Data Page : 0

PDU Format : 254

PDU Specific : 233

Default Priority : 6

Parameter Group Number : 65257 (0xFEE9)

Start Position	Length	Parameter Name	SPN
1-4	4 bytes	Engine trip fuel	182
5-8	4 bytes	Engine total fuel used	250

With ECU simulated parameter value

Engine trip fuel: 0 to 2105540607.5 kg 0.5kg/bit.

Engine total fuel used: 0 to 2105540607.5 kg 0.5kg/bit.

No.6

PGN 61444 ELECTRONIC ENGINE CONTROLLER #1: EEC1

Transmission Repetition Rate : 100ms

Data Length : 8

Reserve Bit : 0

Data Page : 0

PDU Format : 240

PDU Specific : 4

Default Priority : 3

Parameter Group Number : 61444 (0xF004)

Start Position	Length	Parameter Name	SPN
1	4 bits	Status_EEC1	899
2	1 byte	Driver's demand engine - percent torque	512
3	1 byte	Actual engine - percent torque	513
4-5	2 bytes	Engine speed	190
6-8	3 bytes	not defined	

With ECU simulated parameter value

Actual engine - percent torque: -125% to 125% 1%/bit

Engine speed: 0 to 8031.875rpm 0.125rpm/bit

Another position are all 0xFF (not yet implemented)

No.7

PGN 61443 Electronic Engine Controller 2 - EEC2

Transmission Repetition Rate : 50ms

Data Length : 8

Reserve Bit : 0

Data Page : 0

PDU Format : 240

PDU Specific : 3

Default Priority : 3

Parameter Group Number : 61443 (0xF003)

Start Position	Length	Parameter Name	SPN
1.1	2 bits	Accelerator Pedal 1 Low Idle Switch	558
1.3	2 bits	Accelerator Pedal Kickdown Switch	559
1.5	2 bits	Road Speed Limit Status	1437
1.7	2 bits	Accelerator Pedal 2 Low Idle Switch	2970
2	1 byte	Accelerator Pedal Position 1	91
3	1 byte	Engine Percent Load at Current Speed	92
4	1 byte	Remote Accelerator Pedal Position	974
5	1 byte	Accelerator Pedal Position 2	29
6.1	2 bits	Vehicle Acceleration Rate Limit Status	2979

With ECU simulated parameter value

Accelerator Pedal Position 1: 0% to 100% 0.4%/bit

Engine Percent Load at Current Speed: 0% to 125% 1%/bit

Another position are all 0xFF (not yet implemented)

No.8

PGN 65270 Electronic Engine Controller 2 - EEC2

Transmission Repetition Rate : 0.5s

Data Length : 8

Reserve Bit : 0

Data Page : 0

PDU Format : 254

PDU Specific : 246

Default Priority : 6

Parameter Group Number : 65270 (0xFEF6)

Start Position	Length	Parameter Name	SPN
1	1 byte	Particulate trap inlet pressure	81
2	1 byte	Boost pressure	102
3	1 byte	Intake manifold temperature	105
4	1 byte	Air inlet pressure	106

Start Position	Length	Parameter Name	SPN
5	1 byte	Air filter differential pressure	107
6-7	2 bytes	Exhaust gas temperature	173
8	1 byte	Coolant filter differential pressure	112

With ECU simulated parameter value

Boost pressure: 0 to 500kPa (72.5psi) 2kPa/bit

Another position are all 0xFF (not yet implemented)

No.9

PGN 65132 Tachograph - TCO1

Transmission Repetition Rate : 50ms

Data Length : 8

Reserve Bit : 0

Data Page : 0

PDU Format : 254

PDU Specific :108

Default Priority : 6

Parameter Group Number : 65132 (0xFE6C)

Start Position	Length	Parameter Name	SPN
1.1	3 bits	Driver 1 working state	1612
1.4	3 bits	Driver 2 working state	1613
1.7	2 bits	Vehicle motion	1611
2.1	4 bits	Driver 1 Time Related States	1617
2.5	2bits	Driver card, driver 1	1615
2.7	2 bits	Vehicle Overspeed	1614
3.1	4 bits	Driver 2 Time Related States	1618
3.5	4 bits	Driver card, driver 2	1616
4.1	2 bits	System event	1622
4.3	2 bits	Handling information	1621
4.5	2 bits	Tachograph performance	1620
4.7	2 bits	Direction indicator	1619
5-6	2 bytes	Tachograph output shaft speed	1623
7-8	2 bytes	Tachograph vehicle speed	1624

With ECU simulated parameter value

Tachograph vehicle speed: 0 to 250.996km/h 1/256km/h/bit

Another positions are all 0xFF (not yet implemented)

No.10

PGN 65265 Cruise Control/Vehicle Speed

(This PGN is only supported for Version 2.00 of the year 2017 or after)

Transmission Repetition Rate : 100ms

Data Length : 8

Reserve Bit : 0

Data Page : 0

PDU Format : 254

PDU Specific :241

Default Priority : 6

Parameter Group Number : 65265 (0xFEF1)

Start Position	Length	Parameter Name	SPN
1.1	2 bits	Two Speed Axle Switch	69
1.3	2 bits	Parking Brake Switch	70
1.5	2 bits	Cruise Control Pause Switch	1633
1.7	2 bits	Park Brake Release Inhibit Request	3807
2-3	2 bytes	Wheel-Based Vehicle Speed	84
4.1	2 bits	Cruise Control Active	595
4.3	2 bits	Cruise Control Enable Switch	596
4.5	2 bits	Brake Switch	597
4.7	2 bits	Clutch Switch	598
5.1	2 bits	Cruise Control Set Switch	599
5.3	2 bits	Cruise Control Coast (Decelerate) Switch	600
5.5	2 bits	Cruise Control Resume Switch	601
5.7	2 bytes	Cruise Control Accelerate Switch	602
6	1 byte	Cruise Control Set Speed	86
7.1	5 bits	PTO Governor State	976
7.6	3 bits	Cruise Control States	527
8.1	2 bits	Engine Idle Increment Switch	968
8.3	2 bits	Engine Idle Decrement Switch	967
8.5	2 bits	Engine Test Mode Switch	966
8.7	2 bits	Engine Shutdown Override Switch speed	1237

With ECU simulated parameter value

Wheel-Based Vehicle Speed: 0 to 250.996km/h 1/256km/h/bit

Another positions are all 0xFF (not yet implemented)

No.11

PGN 65271 VEHICLE ELECTRICAL POWER

Transmission Repetition Rate : 1s

Data Length : 8

Reserve Bit : 0

Data Page : 0

PDU Format : 254

PDU Specific : 247

Default Priority : 6

Parameter Group Number : 65271 (0xFEF7)

Start Position	Length	Parameter Name	SPN
1	1 byte	Net battery current	114
2	1 bytes	Alternator current	115
3-4	2 bytes	Alternator potential (voltage)	167
5-6	2 bytes	Electrical potential (voltage)	168
7-8	2 bytes	Battery potential (voltage), switched	158

With ECU simulated parameter value

Electrical potential (voltage): 0 to 3212.75V 0.05V/bit.

Another positions are all 0xFF (not yet implemented)

No.12

PGN 65266 FUEL ECONOMY

Transmission Repetition Rate : 100ms

Data Length : 8

Reserve Bit : 0

Data Page : 0

PDU Format : 254

PDU Specific : 242

Default Priority : 6

Parameter Group Number : 65266 (0xFEF2)

Start Position	Length	Parameter Name	SPN
1-2	2 byte	Fuel rate	183
3-4	2 bytes	Instantaneous fuel economy	184
5-6	2 bytes	Average fuel economy	185
7-8	2 bytes	not defined	

With ECU simulated parameter value

Fuel rate: 0 to 3212.75L/h 0.05L/h/bit

Another positions are all 0xFF (not yet implemented)

No.13

PGN 65263 ENGINE FLUID LEVEL/PRESSURE

Transmission Repetition Rate : 0.5s

Data Length : 8

Reserve Bit : 0

Data Page : 0

PDU Format : 254

PDU Specific : 239

Default Priority : 6

Parameter Group Number : 65263 (0xFEEF)

Start Position	Length	Parameter Name	SPN
1	1 byte	Fuel delivery pressure	94
2	1 byte	Not defined	
3	1 byte	Engine oil level	98
4	1 byte	Engine oil pressure	100
5-6	2 bytes	Crankcase pressure	101
7	1 byte	Coolant pressure	109
8	1 byte	Coolant level	111

With ECU simulated parameter value

Fuel delivery pressure: 0 to 1000kPa 4kPa/bit

Engine oil pressure: 0 to 1000kPa 4kPa/bit

Coolant pressure: 0 to 500kPa 2kPa/bit

Another position are all 0xFF (not yet implemented)

No.14

PGN 65253 Engine Hours, Revolutions

Transmission Repetition Rate :on request

Data Length : 8

Reserve Bit : 0

Data Page : 0

PDU Format : 254

PDU Specific : 229

Default Priority : 6

Parameter Group Number : 65253 (0xFEE5)

Start Position	Length	Parameter Name	SPN
1-4	4 bytes	Total engine hours	247
5-8	4 bytes	Total engine revolutions	249

With ECU simulated parameter value

Total engine hours: 0 to 210554060.75h 0.05h/bit.

Total engine revolutions: 0 to 4211081215000r 1000r/bit.

No.15

PGN 65214 ELECTRONIC ENGINE CONTROLLER #4: EEC4

Transmission Repetition Rate : on request

Data Length : 8

Reserve Bit : 0

Data Page : 0

PDU Format : 254

PDU Specific : 190

Default Priority : 7

Parameter Group Number : 65214 (0xFEBE)

Start Position	Length	Parameter Name	SPN
1-2	2 byte	Rated engine power	166
3-4	2 bytes	Rated engine speed	189
5-8	4 bytes	not defined	

With ECU simulated parameter value

Rated engine speed: 0 to 8031.875rpm 0.125rpm/bit.

Another positions are all 0xFF (not yet implemented)

No.16

PGN 65248 Vehicle Distance

Transmission Repetition Rate : 100ms

Data Length : 8

Reserve Bit : 0

Data Page : 0

PDU Format : 254

PDU Specific : 224

Default Priority : 6

Parameter Group Number : 65248 (0x00FEE0)

Start Position	Length	Parameter Name	SPN
1-4	4 bytes	Trip Distance	244
5-8	4 bytes	Total Vehicle Distance	245

With ECU simulated parameter value

Trip Distance: 0 to 526,385,151.9 km 0.125 km/bit.

Total Vehicle Distance: 0 to 526,385,151.9 km 0.125 km/bit.

6.2 Setup J1939 Simulator

Let's setup this device simulator.

1 Please plug Simulator USB into your computer, and Run CANSimulator.exe, and then click Menu item: " Simulator Settings ". The following dialog occurs:

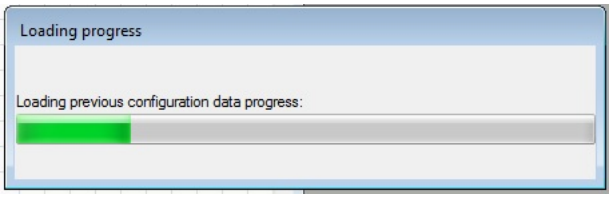


Fig. 57 loading previous settings

2 After loading settings, it will display below:

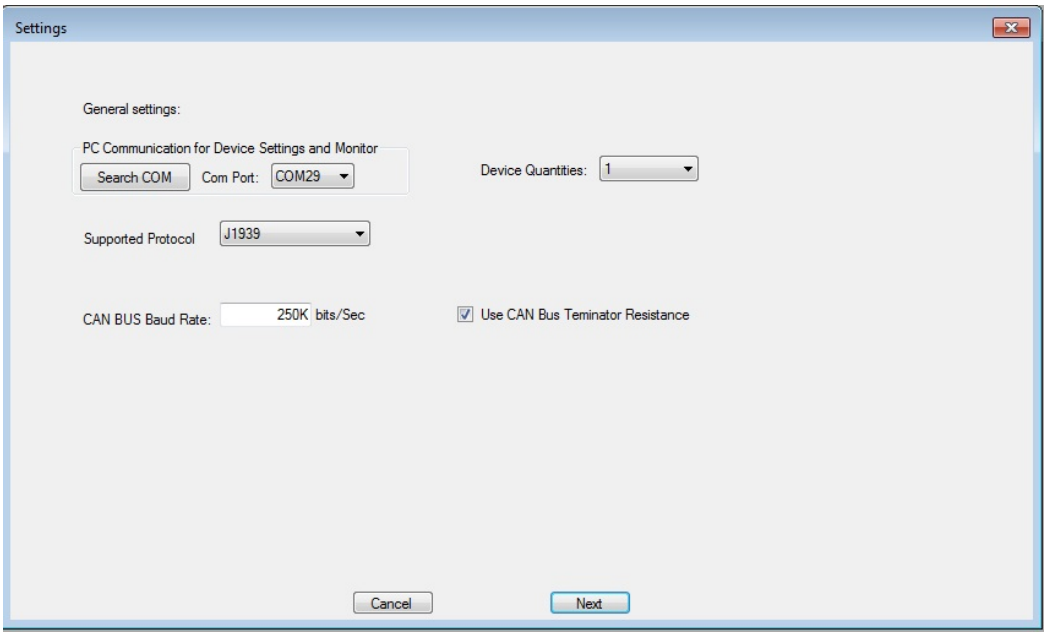


Fig.58 Protocol settings

We click "Search COM" button to select COM Port number from drop list, which is used for CAN Bus simulator. Select Device quantities to 1, and Supported protocol to "J1939". CAN Bus baud rate is 250K. You can enable/disable 120 ohms terminator.

Click "Next" button to next step. Or "Cancel" button to exit settings (cancel all settings you made)

3 After above "Next" button click, the following window will display:

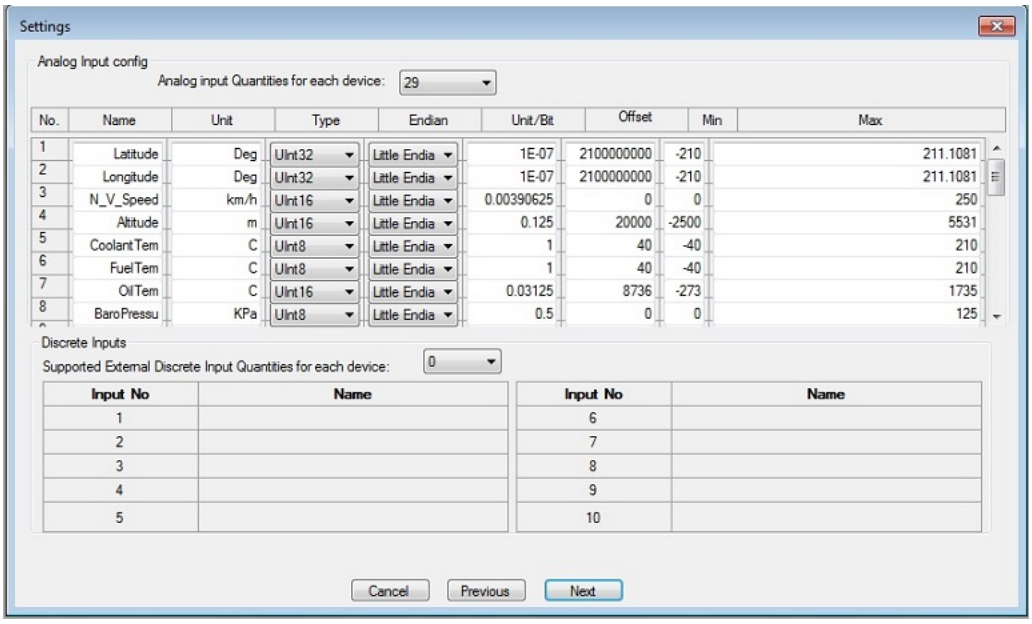


Fig.59 Analog/Digital Input settings

Total Analog inputs qty is 29, we set every Analog inputs as we described in the device description of analog. No any digital inputs. Click "Next" to enter next step or "Previous" to return step 2 or "Cancel" button to exit settings (cancel all settings you made).

Notes: *If you simulate multiple devices, they will have the same analog inputs (or Digital inputs). Only values are different.*

4 In this step, you will set faults/warnings as display below:

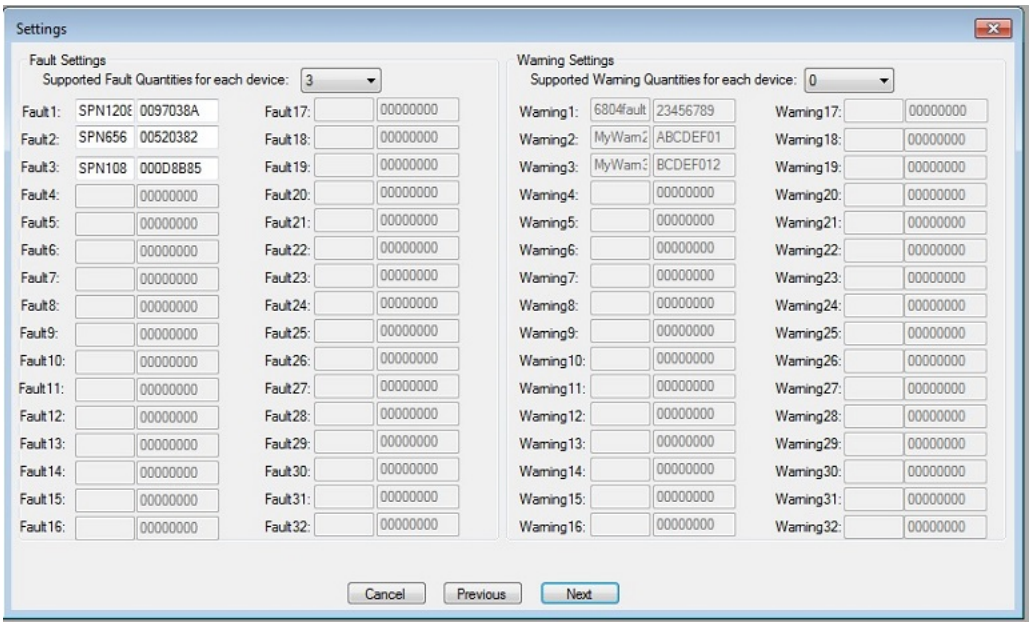


Fig. 60 Faults/warnings settings

We set 3 faults: SPN1208, SPN656, and SPN108. There are 32 bits of Hex DTC on each fault right side. Above DTCs are Format version 1. If other DTC Format versions are used , please use other values as we write in device description. Click "Next" to enter next step or "Previous" to return step 3 or "Cancel" button to exit settings (cancel all settings you made).

Notes*: If you simulate multiple devices, they will have the same faults (or warnings).*

5. In this step, we will set up my device J1939 address, VIN# and 8 bytes of Device name, and supported PGNs as shown as below.

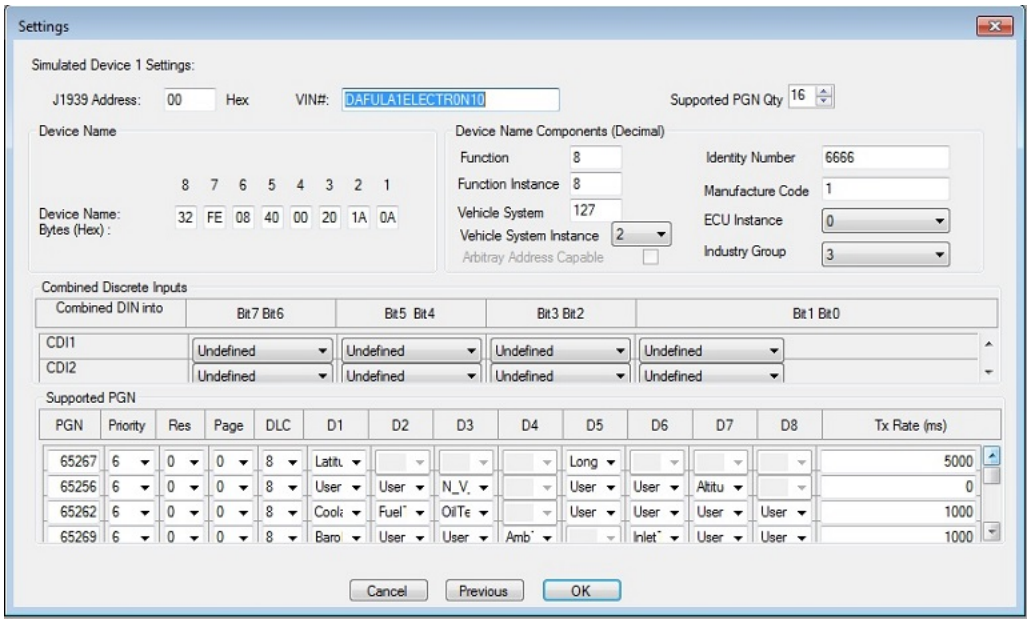


Fig. 61 Device PGN Settings

J1939 address is 0 , supported PGN Qty is 16 (It is the number we described in device description).
16 PGNs Data settings are the same as described in device description.
Click "OK" button to finish all settings. Or "Previous" to return step 5, "Cancel" button to exit settings (cancel all settings you made)

***Notes**: If you have *. dcfg settings file, you can skip 1 to 5, just use File/Use settings File .....*

6 For J1939 Simulator, we can let state machine empty (Just click menu "File/New State Machine").
Now, you open Analog inputs windows/ monitor CAN window/Fault window as picture below:

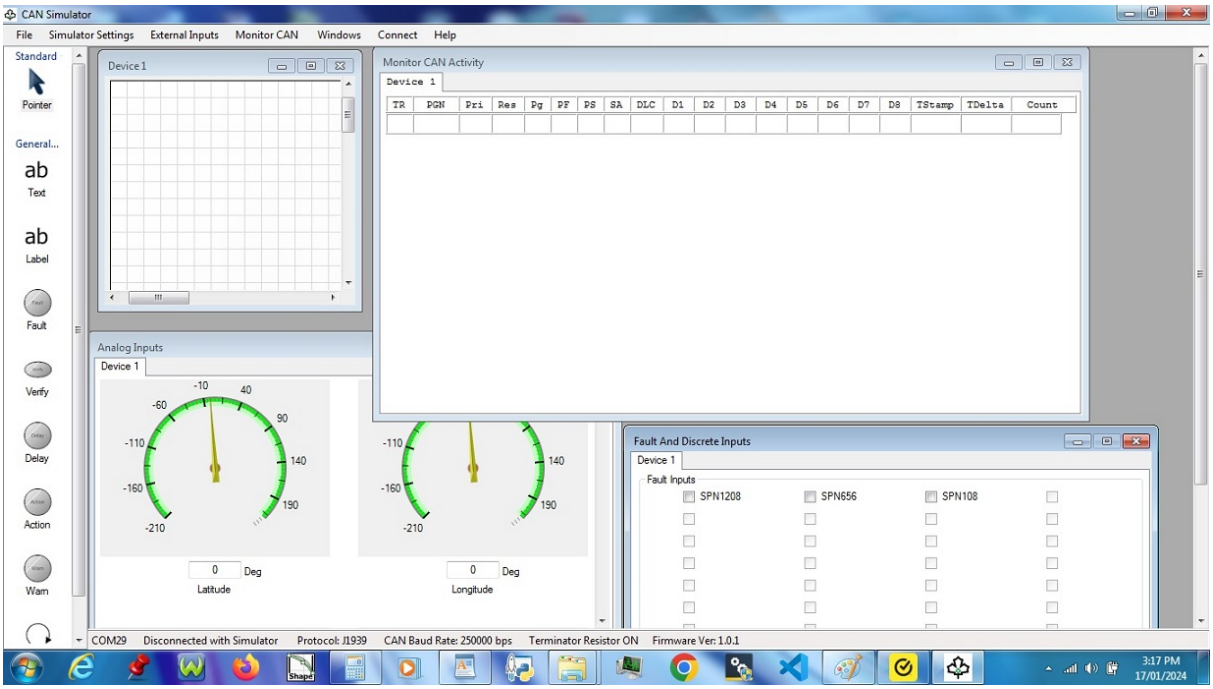


Fig. 62 open windows

Please click menu "Connect" to start J1939 Simulator, you will see simulator result as picture below:

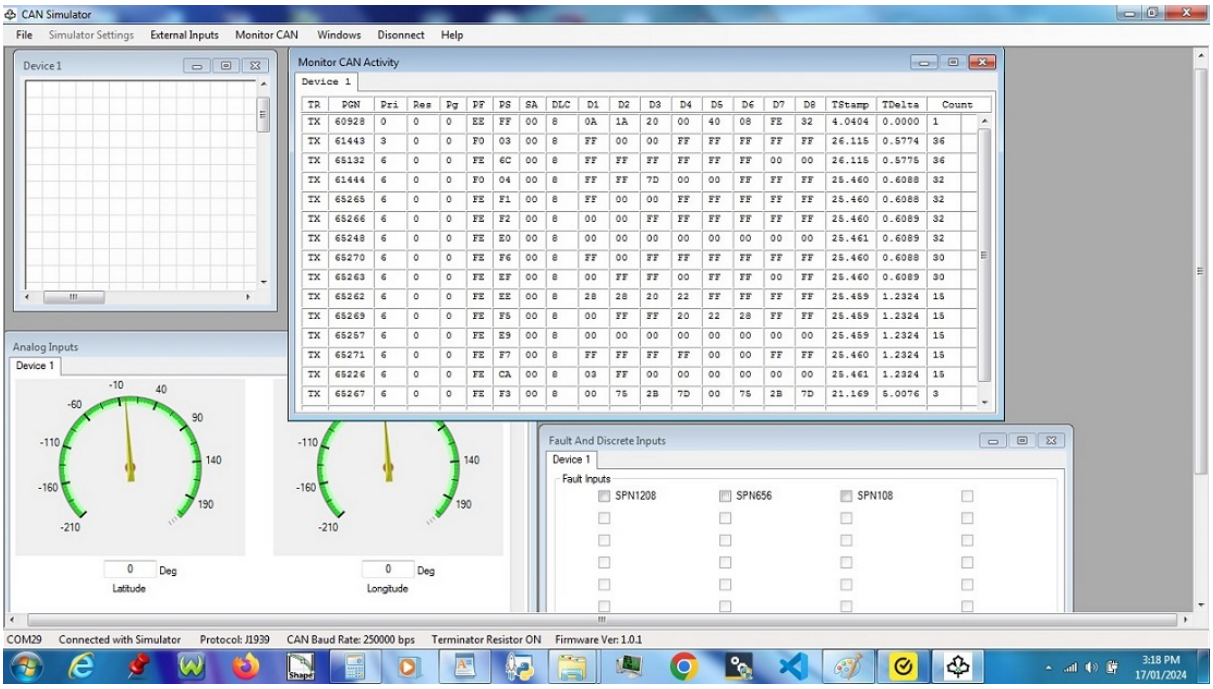


Fig. 63 Simulator is running

This example files are in folder: Top folder\CANbusSim\Example\J1939_1

If we want the following analogs value calculated according to run time:

- The 11th analog: Engine trip fuel (Inside PGN 65257 FUEL CONSUMPTION). We give the value according to "Fuel rate"
- The 12th analog: Engine total fuel used (Inside PGN 65257 FUEL CONSUMPTION). We give the value according to "Fuel rate"
- The 25th analog: Total engine hours (Inside PGN 65253 Engine Hours, Revolutions). We give the value according to actual running time.
- The 26th analog: Total engine revolutions (Inside PGN 65253 Engine Hours, Revolutions) We give the value according to engine speed and actual running time.
- The 28th analog: Trip Distance (Inside PGN 65248 Vehicle Distance). We give the value according to tachograph vehicle speed and actual running time.
- The 29th analog: Total Vehicle Distance (Inside PGN 65248 Vehicle Distance). We give the initial value 100,000km when Simulator power on, and then we will accumulate the distance according to tachograph vehicle speed and actual running time.

We can use 23 Analog inputs instead of 29 Analog inputs , the decreased 6 analog inputs are calculated in run time by state machine VB script.
And for supported PGN data field must be changed to "User defined"
This example files are in folder: Top folder\CANbusSim\Example\J1939_2

7 One Example for CANopen

7.1 CANopen Server Device description

We will demonstrate one CANopen Server device with Node ID=1 and CAN baud rate =250Kbit/sec.

Device has 2 Analog inputs below:

- The 1st Analog:
  Name: An1, Unit: V, Type: int16, scale=1, range: 0 to 1000
- The 2nd Analog:
  Name: An2, Unit:V, Type:float (4 bytes) , range: 0 to 1000.00

Device has 10 Digital inputs, names are from DIN1 to DIN10
Device supports 4 Faults and 1 warning as below:

- The 1st Fault:
  Name: F1, Error Code Plus additional information ="000A1001" (Error code is in LSW), Error

Register="07"

- The 2nd Fault:
  Name: F2, Error Code Plus additional information ="000B1002" (Error code is in LSW), Error Register="09"
- The 3rd Fault:
  Name: F3, Error Code Plus additional information ="000C1003" (Error code is in LSW), Error Register="21"
- The 4th Fault:
  Name: F4, Error Code Plus additional information ="000D1004" (Error code is in LSW), Error Register="03"
- The 1st Warning:
  Name: w1, Error Code Plus additional information ="0001100A" (Error code is in LSW), Error Register="A9"

Device supports 2 TPDOs as below:

- The 1st TPDO:
  Hardware implement. DLC is 8 (Every hardware implement TPDO has fixed DLC=8). COB-ID=0x181, It is Async and Cyclic. Cycle period is the item in dictionary with index=2002 and subindex=1.
  Data1 maps to the item in dictionary with index=3000 and subindex=1, which is DIN1 to DIN8 from LSb to MSb.
  Data2 only uses 2 bits of the most right-side (Bit0 and Bit1). Bit0 maps to the item in dictionary with index=3000 and subindex=2, which is DIN9. Bit1 maps to the item in dictionary with index=3000 and subindex=3, which is DIN10.
- The 2nd TPDO:
  Software implement. DLC is 6. COB-ID=0x281, It is Async and Cyclic. Cycle period is the item in dictionary with index=2002 and subindex=2.
  Data1 and Data2 map to the item in dictionary with index=3001 and subindex=1, which is An1 (Little Endian).
  Data3 to Data6 map to the item in dictionary with index=3001 and subindex=2, which is An2 (Little Endian).

Device supports 2 RPDOs as below:

- The 1st RPDO:
  COB-ID=0x201, It is Async and Acyclic.
  Data1 and Data2 maps to the item in dictionary with index=2002 and subindex=1, which is cycle period of TPDO1. (Little Endian). So this RPDO can change cycle period of TPDO1 on run time.

- The 2nd TPDO:

  COB-ID=0x301, It is Async and Acyclic.

  Data1 and Data2 maps to the item in dictionary with index=2002 and subindex=2, which is cycle period of TPDO2. (Little Endian). So this RPDO can change cycle period of TPDO2 on run time.

We can use state machine to implement device behaviour. Of cause, if you don't simulate special behaviour, you don't need State machine.

Our simulated device behaviour as below:

1. Wait for RPDO1. If receiving RPDO1, it will delay 5 seconds and enter "Action" State.
2. In action state, it will write 100 to Debug(0).
3. If fault occurs, it will enter "Fault" State.
4. In Fault state, it will write 10.00 to DebugSng(0).
5. If fault disappears, go to step 1.

7.2 Setup CANopen Server Simulator

Let's setup this device simulator.

1 Please plug Simulator USB into your computer, and Run CANSimulator.exe, and then click Menu item: " Simulator Settings ". The following dialog occurs:

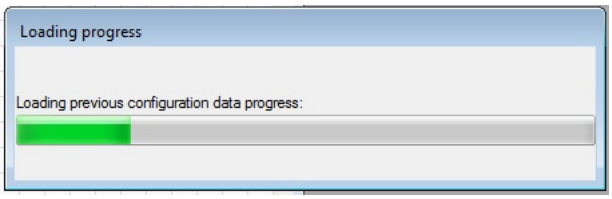


Fig. 64 loading previous settings

2 After loading settings, it will display below:

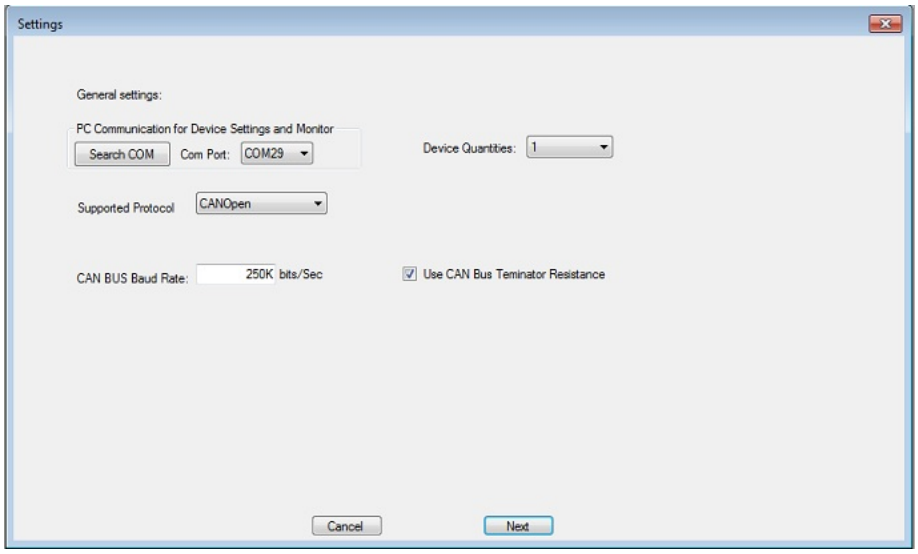


Fig. 65 Protocol settings

We click "Search COM" button to select COM Port number from drop list, which is used for CAN Bus simulator. Select Device quantities to 1, and Supported protocol to "CANOpen". CAN Bus baud rate is 250K. You can enable/disable 120 ohms terminator.
Click "Next" button to next step. Or "Cancel" button to exit settings (cancel all settings you made)

3 After above "Next" button click, the following window will display:

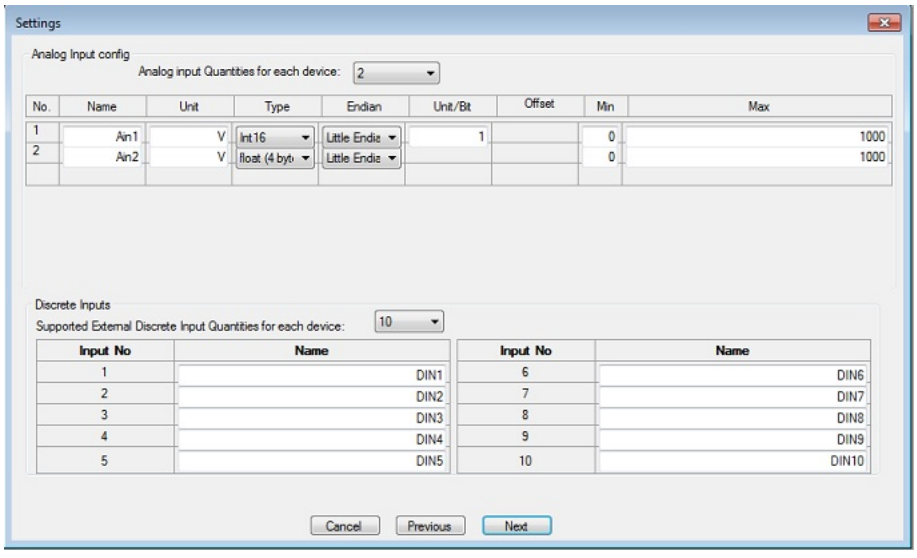


Fig.66 Analog/Digital Input settings

Total Analog inputs qty is 2, we set every Analog inputs as we described in the device description of analog. Total digital inputs qty is 10. And digital inputs names are from DIN1 to DIN10. Click "Next" to enter next step or "Previous" to return step 2 or "Cancel" button to exit settings (cancel all settings you made).

Notes: *If you simulate multiple devices, they will have the same analog inputs and Digital inputs. Only values are different.*

4 In this step, you will set faults/warnings as display below:

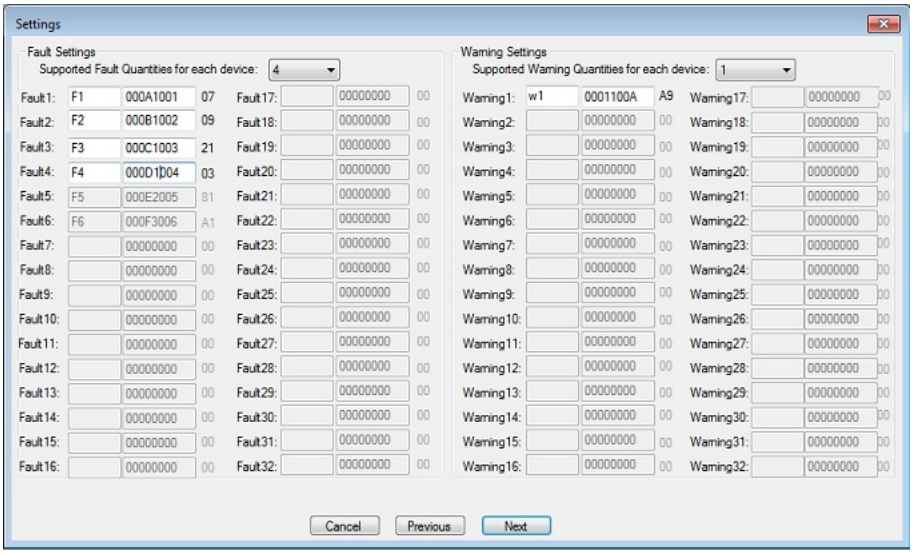


Fig. 67 Faults/warnings settings

We set 4 faults/1 Warning: F1, F2, F3 , F4, and w1. There are combination from 16 bits of "Error code" plus 16 bits of "additional information" (LSW is 16 bits' Error code) on each fault/warning name's right side. And after this right side, it is 8 bits' Error register. You just double click "Error register" data , a pop-up window will occur below:

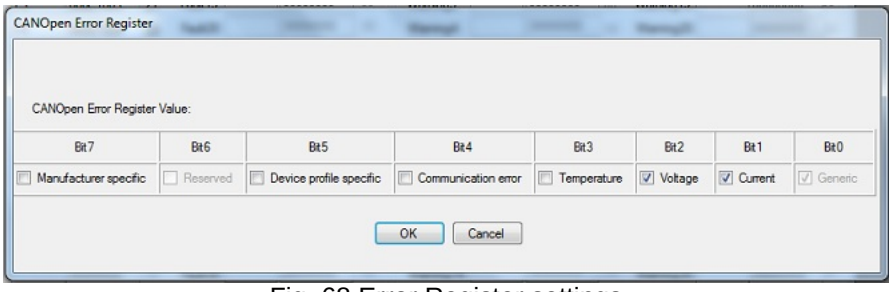


Fig. 68 Error Register settings

For 4 faults and 1 Warning, the "Error code" plus "additional information" will be "000A1001", "000B1002", "000C1003", "000D1004" and "0001100A", and the Error register will be "07","09","21","03", and "A9".

Notes*: If you simulate multiple devices, they will have the same faults and warnings.*

5. In this step, we will set up my device node ID and TPDO and RPDO as shown as below.

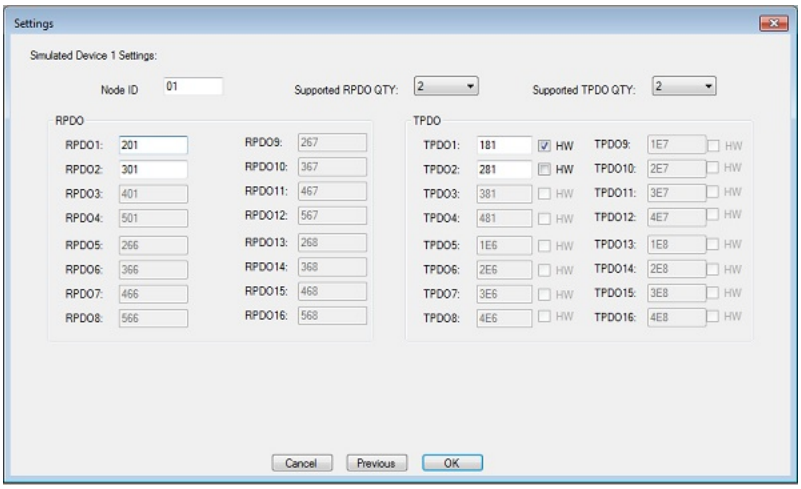


Fig. 69 CANOpen Device settings

Beside TPDO COB-ID, there is check box called "HW" which means hardware implement TPDO if checked. The total max Qty for hardware TPDO from all devices is " 8 minus device Qty" because our hardware has Max 8 hardware cyclic transmitted CAN Bus frames. Heart Beat must be hardware cyclic transmitted CAN Bus frame.

***Notes**: If you have *. dcfg settings file, you can skip 1 to 5, just use File/Use settings File .....*

6. For CANOpen Simulator, you must set Object Dictionary. There are 6 different dictionary areas. The first area is "Communication Profile", the 2nd area is "RPDO Communication parameter", the 3rd area is "RPDO Mapping Parameter", the 4th area is "TPDO Communication parameter", the 5th area is "TPDO Mapping Parameter", the 6th area is "Manufacturer specific profile". You can choose any one from menu "Object Dictionary".

Please click menu "Object Dictionary"/"Communication Profile". The following window occurs:

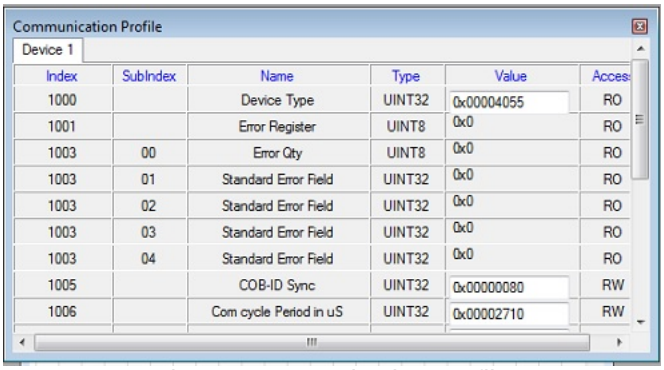


Fig. 70 Communication Profile

The 1st column is index, it is cannot be edited because this is standard index for "Communication

Profile". The 5th column is dictionary item value, some is editable , which means you can configure its value. The 6th column is "Access property" , it can be "RO" (Read-only") or "RW" (Read-write). If one item is "RO", it means that you can use SDO to read this item, but you cannot change its value. If one item is "RW", it means that you can use SDO to read/Write this item.

We support SDO expedited transfer for read/write and SDO Segmented transfer for read.

Please click menu "Object Dictionary"/"RPDO Communication parameter". The following window occurs:

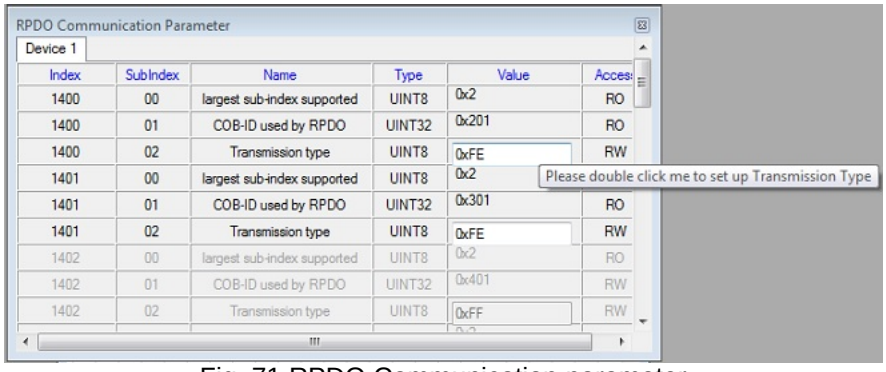


Fig. 71 RPDO Communication parameter

There is only "Transmission type" item which is configurable. Please double click on Transmission type's value. it will pop-up dialog below:

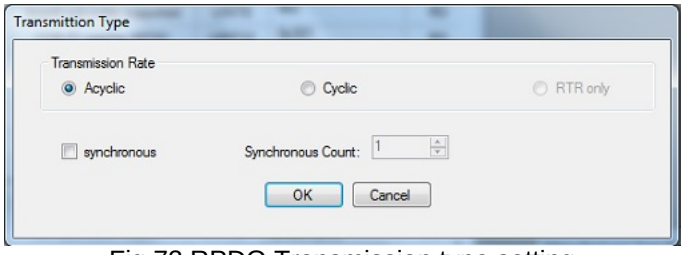


Fig.72 RPDO Transmission type setting

Please set it as Fig.72. Please click menu "Object Dictionary"/"RPDO Mapping Parameter". The following window occurs:

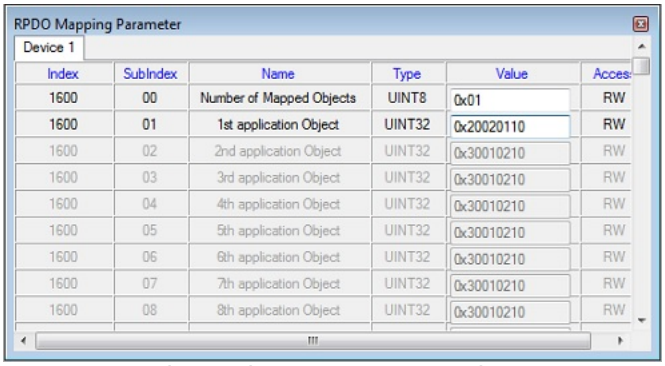


Fig.73 The 1st RPDO Mapping

We set the 1st RPDO mapping as Fig.73, and the 2nd RPDO mapping as Fig.74.

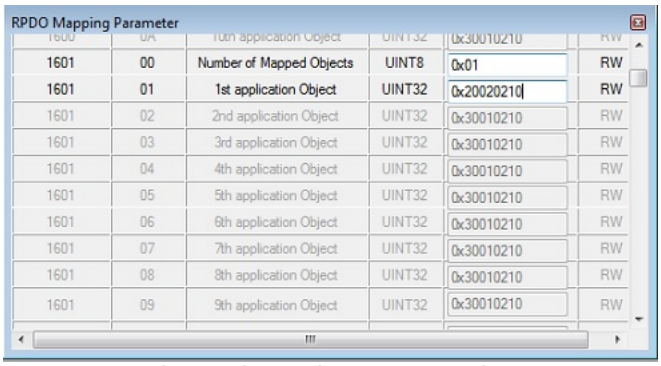


Fig.74 The 2nd RPDO Mapping

Please click menu "Object Dictionary"/"TPDO Communication parameter". The following window occurs:

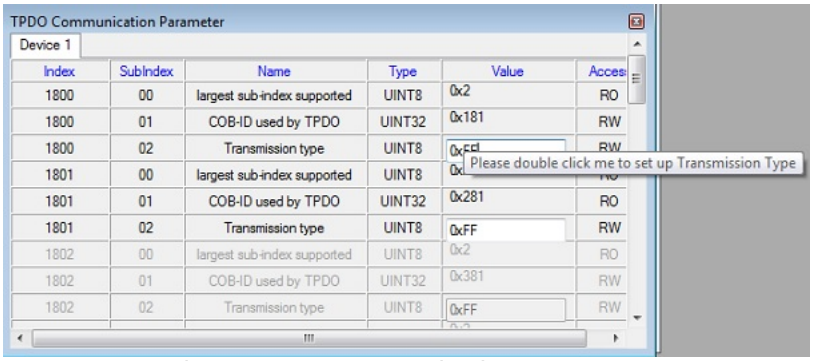


Fig.75 TPDO Communication parameter

There is only "Transmission type" item which is configurable. Please double click on Transmission type's value of the 1st TPDO. it will pop-up dialog below:

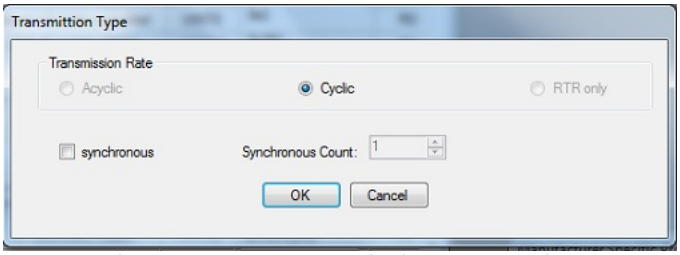


Fig.76 TPDO1 Transmission type setting

We know that TPDO1 is hardware implemented TPDO in device settings. All Hardware TPDO cannot be RTR-only. So RTR-only is grey. Please set it as Fig.76 for TPDO1's transmission type.
The 2nd TPDO is software implemented TPDO, We can set it as TPDO1.

***Notes**: For TPDOs, if anyone TPDO transmission type is Synchronous. all other TPDO will be Synchronous because our CAN BUS hardware uses hardware Synchronous. For our hardware platform, if synchronous transmit, All transmit frame will do actually transmit when it received synchronous CAN Bus frame. Even though SDO transmits when*

operation state, outside CANOpen master must send Synchronous CAN BUS frame in operation state.

Please click menu "Object Dictionary"/"TPDO Mapping Parameter". The following window occurs:

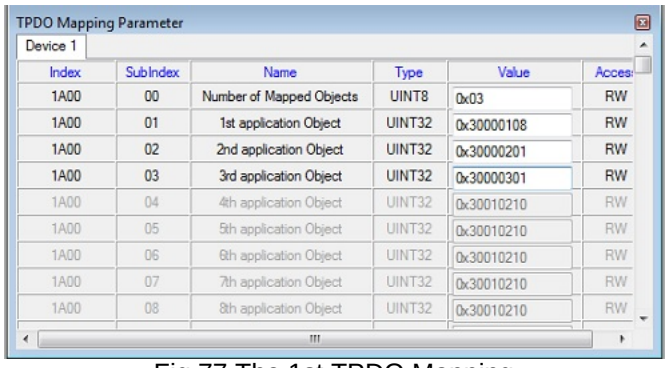


Fig.77 The 1st TPDO Mapping

We set the 1st TPDO mapping as Fig.77, and the 2nd TPDO mapping as Fig.78.

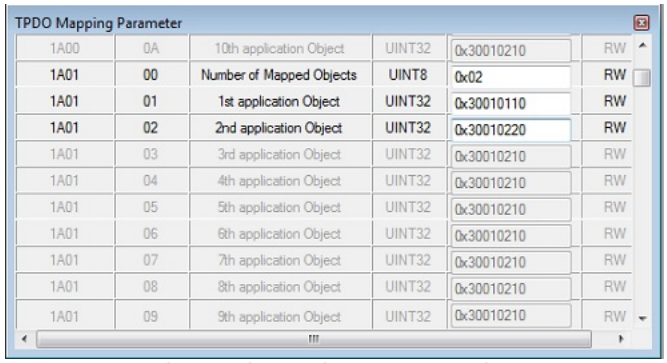


Fig.78 The 2nd TPDO Mapping

Please click menu "Object Dictionary"/"Manufacturer specific profile". The following window occurs:

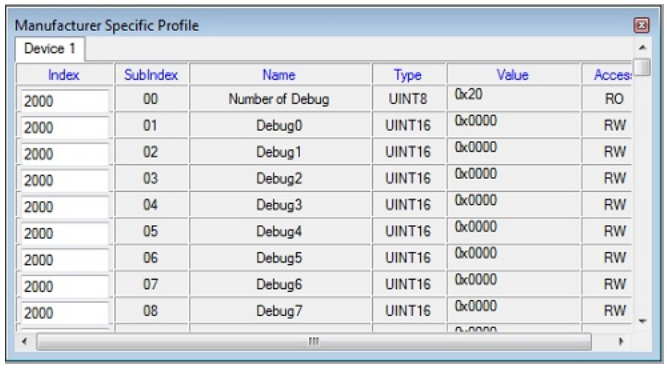


Fig.79 Manufacturer specific profile

The 1st column is index, it is can be edited for "Manufacturer specific Profile". But it must be range "2000" to "5FFF". The 5th column is dictionary item value, some is editable , which means you can configure its value. The 6th column is "Access property" , it can be "RO" (Read-only") or "RW" (Read-write). If one item is "RO", it means that you can use SDO to read this item, but you cannot change its value. If one item is "RW", it means that you can use SDO to read/Write this item.

We have 5 groups of items for "Manufacturer specific Profile". The first group is 32 debug registers which is 16 bits' unsigned integer. Please see Fig.79. You can use SDO to read/write debug register.

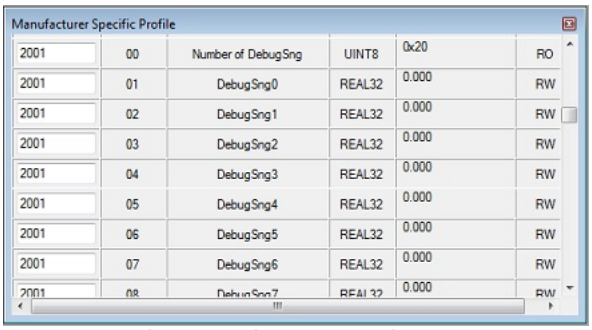


Fig. 80 DebugSng registers

The 2nd group is 32 debugSng registers which is 4 bytes of float number. Please see Fig.80. You can use SDO to read/write debug register.

The 3rd group is 16 TPDO's Cycle periods which is 16 bits' unsigned integer, unit is ms. Please see Fig.81. You can use SDO to read/write debug register.

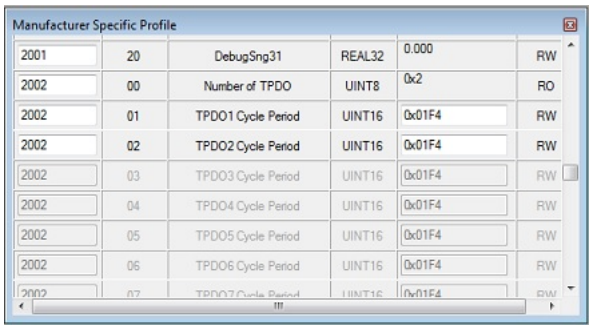


Fig.81 TPDO Cycle period

The 4th group is Digital Inputs. Please see Fig.82 below:

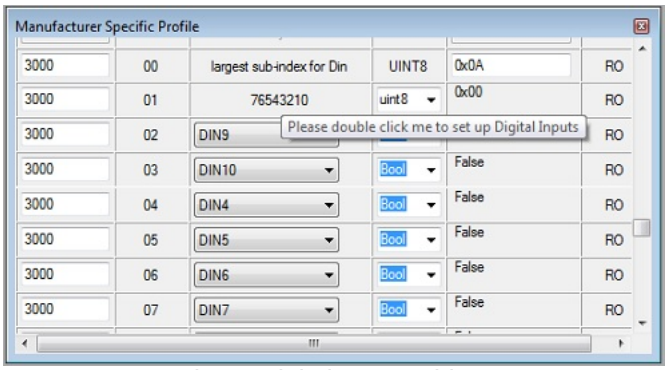


Fig.82 Digital Inputs object

For Digital input object, if data type is bool, it will be one digital input. You can select which digital input from drop list. However, when data type is "uint8" or "Uint16" , this one digital input object will

contains multiple digital inputs. Please double click on the 3rd column, the following pop-up dialog will occur:

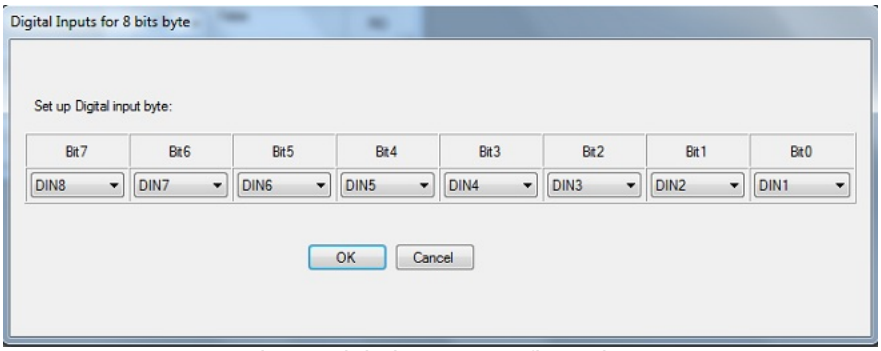


Fig.83 Digital Inputs configuration

You can configure every bit as one digital inputs as Fig.83. And the 3rd column will display 8 or 16 digital numbers. This number is digital input number minus 1. For example "76543210" , from right to left , Bit 0 is 0, it means that bit0 of uint8 is Digital input 1. Bit7 is 7, it means that bit7 of uint8 is Digital input 8.

The 5th group is Analog Inputs. Please see Fig.84 below:

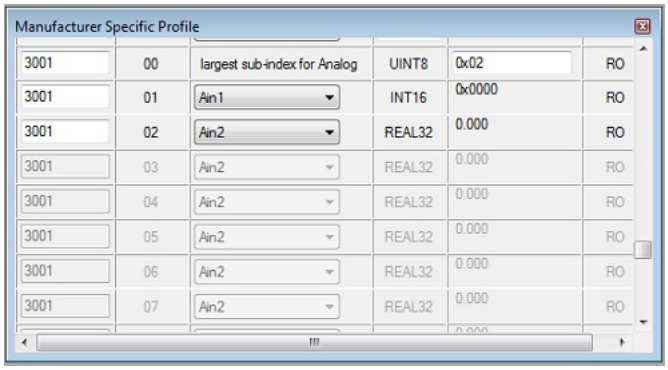


Fig.84 Analog inputs configuration

You only need to select which analog input from drop list.

***Notes**: If you have *. od dictionary file, you can skip 6, just use File/Use OD File ..*

7. For CANopen Simulator, we can let state machine empty. However, if you want to set up special behaviour of device, you can set up state machine, Please click menu "File/New State Machine", and use left tools drag/drop to set up state machine as below:

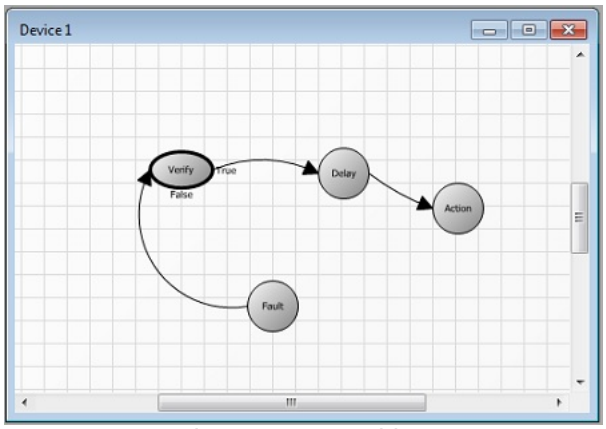


Fig.85 State machine

Make "Verify State" to be Default State (It has thick border). Double click "Verify State", we will type our VBScript as Fig.86 below:

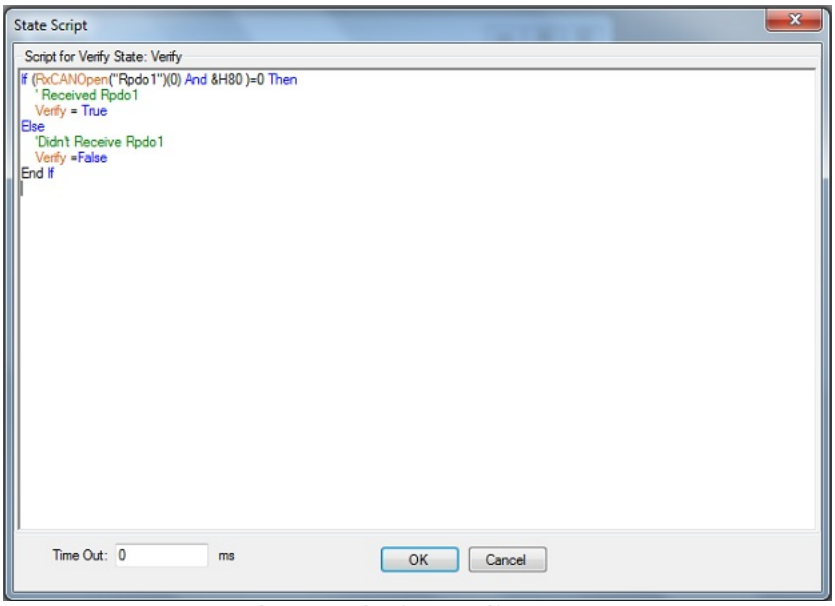


Fig.86 Script for "Verify" State

In "Verify" state, it will see whether to receive RPDO1. If receiving RPDO1, it will make "Verify variable "= True, and transfer to "Delay" state. Double click "Delay State", we will type "5000" in pop-up Script dialog window as Fig.87 below:

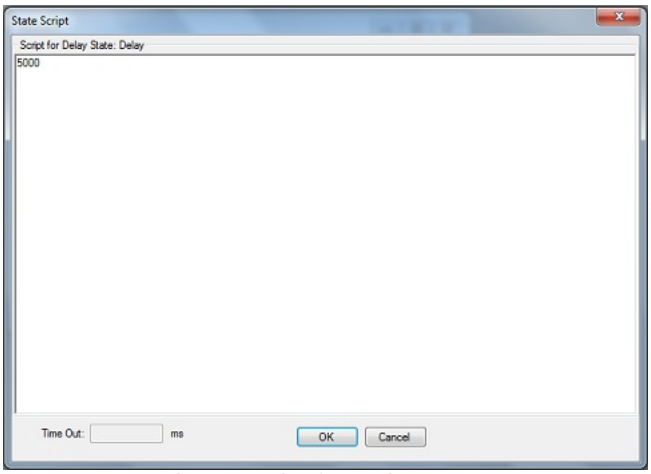


Fig.87 Script for Delay State

It means that system will delay 5000ms and enter next state "Action". Double click "Action State", we will type our VBScript in pop-up Script dialog window as Fig.88 below:

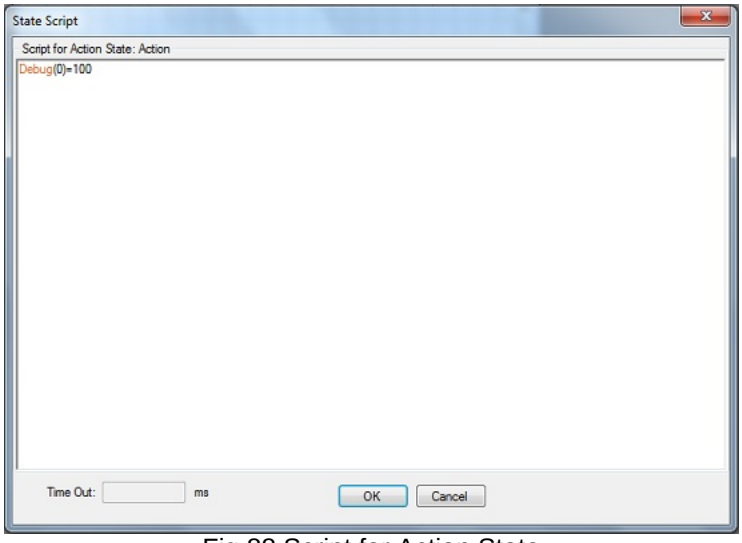


Fig.88 Script for Action State

In "Action" state, Debug[0] will be assigned 100. Double click "Fault State", we will type our VBScript in pop-up Script dialog window as Fig.89 below:

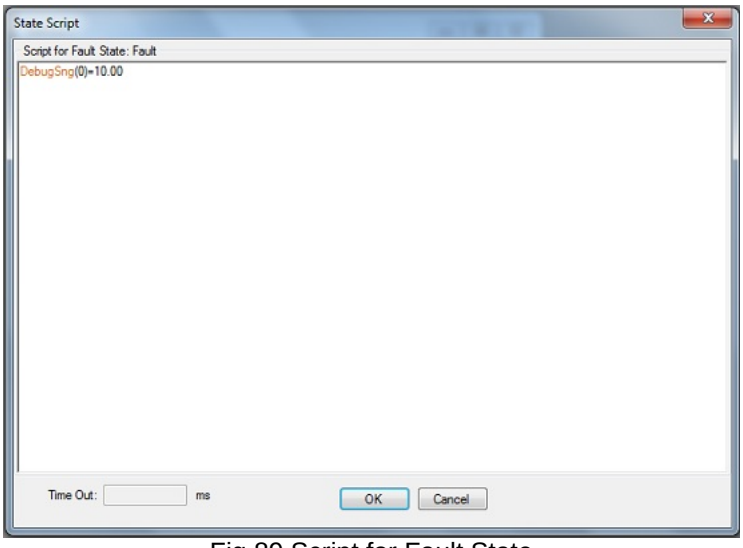


Fig.89 Script for Fault State

In "Fault" state, DebugSng[0] will be assigned 10.00.

***Notes**: If you have *.nsprj *.scr state machine file, you can skip 7, just use File/Open State Machine .....*

8. Now, you have done all device settings and script, please click menu "Connect" to start CANopen Simulator, you will see simulator result as picture below::

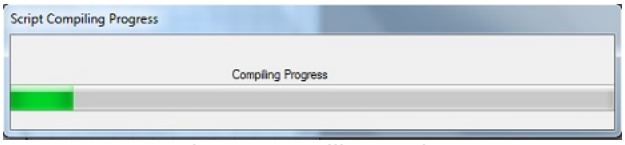


Fig.90 Compiling Script

If no any error for compiling, you can open "Analog inputs", "Fault And Discrete Inputs", "Monitor CAN", and "Object Dictionary" windows, you will see screen like Fig.91 below:

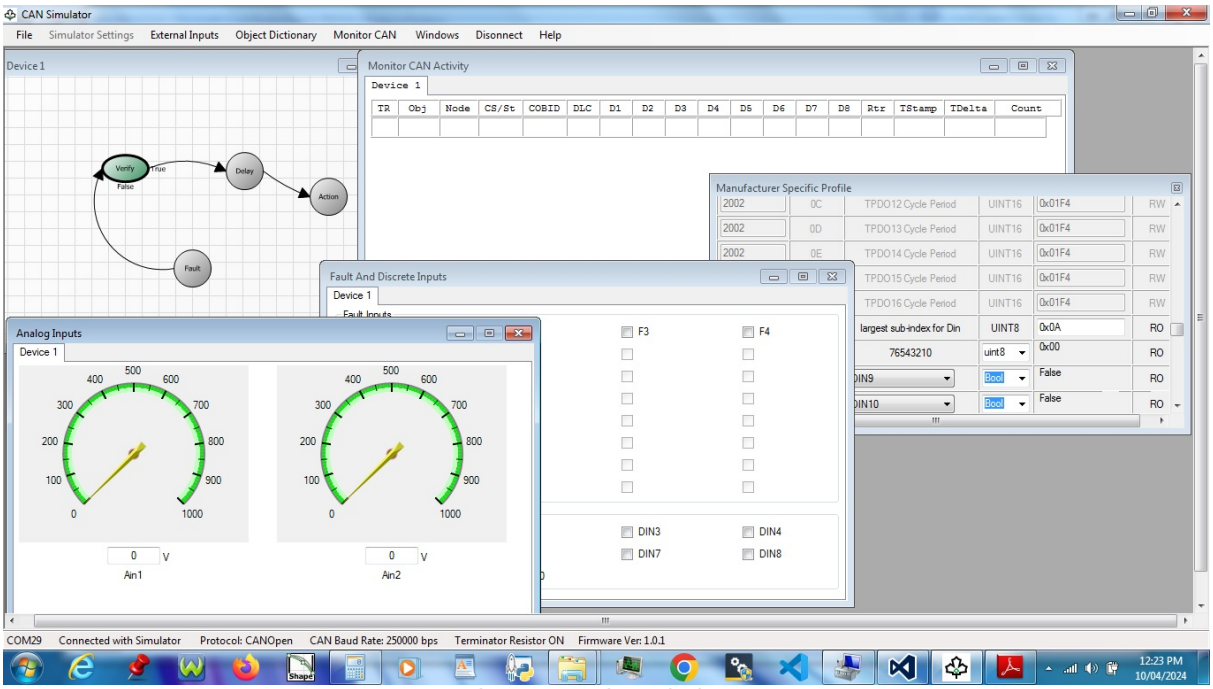


Fig.91 Running windows

We will see State is in "Verify" State. However, we will see nothing in "Monitor CAN Activity" window. It should have "Heartbeat" frame to display "Pre-Operation" State. We know "Heartbeat" implemented in Hardware, we cannot see it in "Monitor CAN Activity" window. We used Another "CAN Bus Analyzer" from Microchip Company. (You can use our CAN Bus Analyzer too. But in order to display result with full confidence, we use the 3rd party's CAN BUS Analyzer) The result is in Fig.92 below:

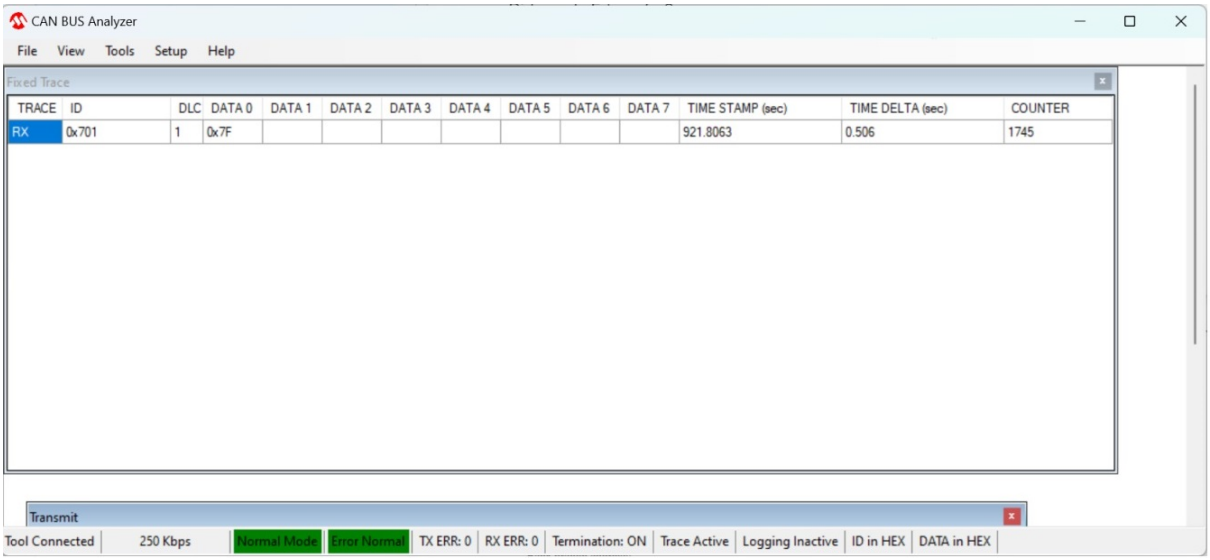


Fig.92 Heart Beat

Indeed, we see the heart beat with value=0x7F(Pre-Operation) and cycle period=0.5 Second. Now we

use Microchip CAN BUS Analyzer to send NMT, let Device enter "Operation" State, Please see Red color frame in Fig.93 below:

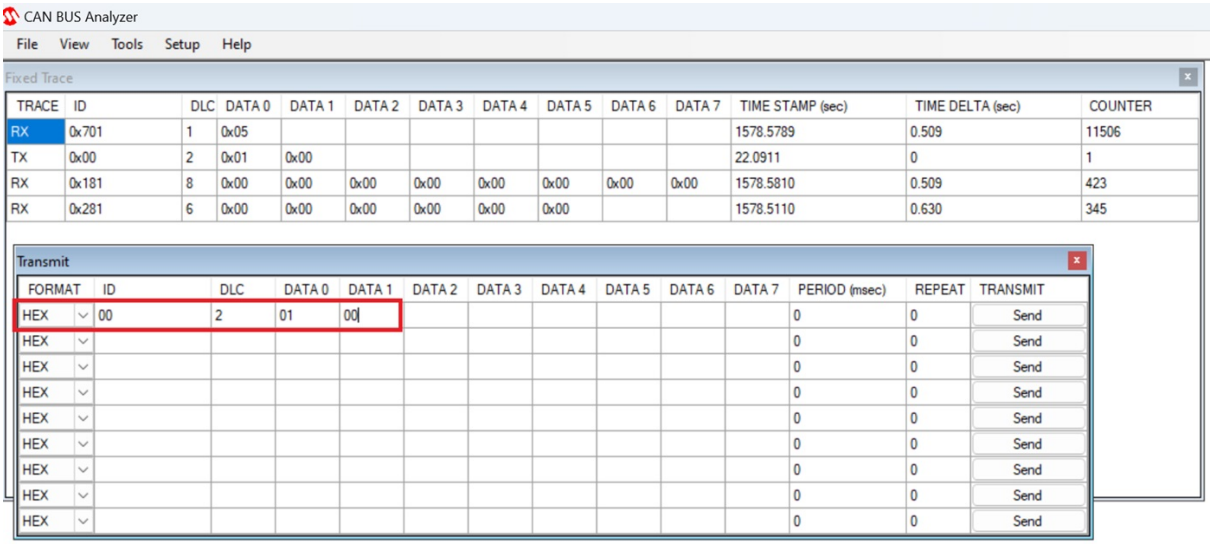
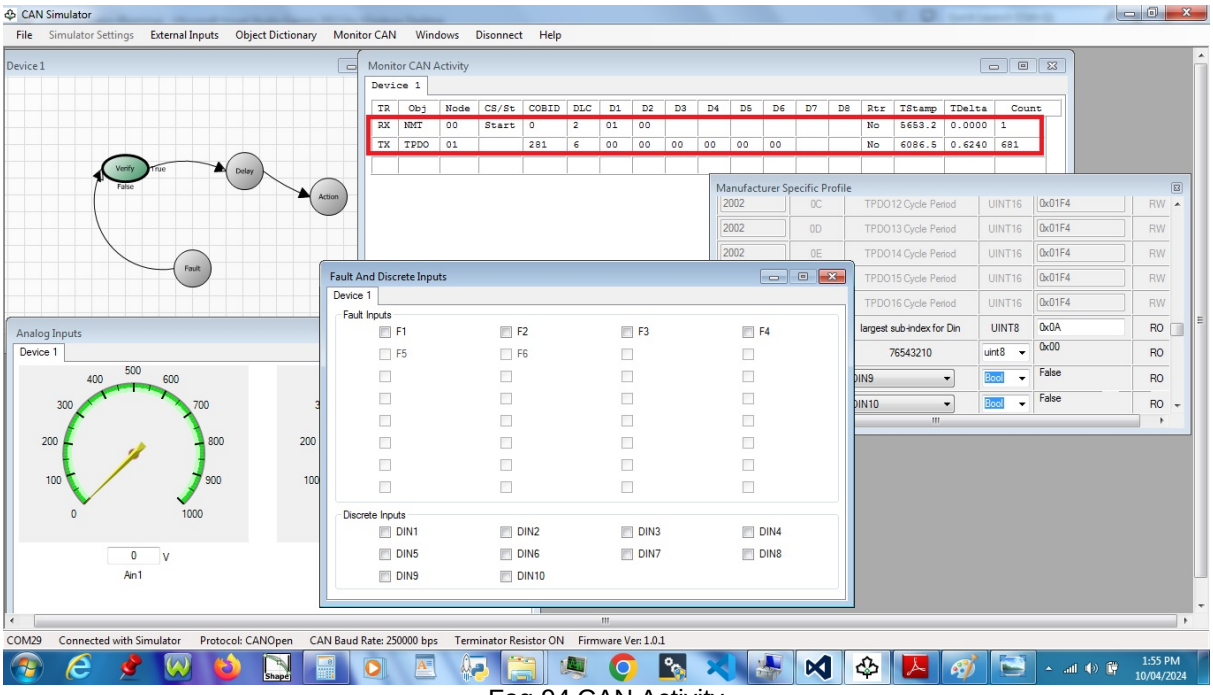


Fig.93 Send NMT once to enter "Operation"

In Fig.93, we see "Heart beat" with data=0x05 (Operation State) and TPDO1 and TPDO2. What do we see in our "Monitor CAN Activity" window? Please see Fig.94 in red color. We only see NMT we received and TPDO2 we sent. We didn't see TPDO1 and Heart Beat. This is because those items are hardware implemented. We cannot see any items which is hardware implemented as we said before.



Fog.94 CAN Activity

We Change DIN1= true, DIN2=true, DIN9=true as Fig.95 below:

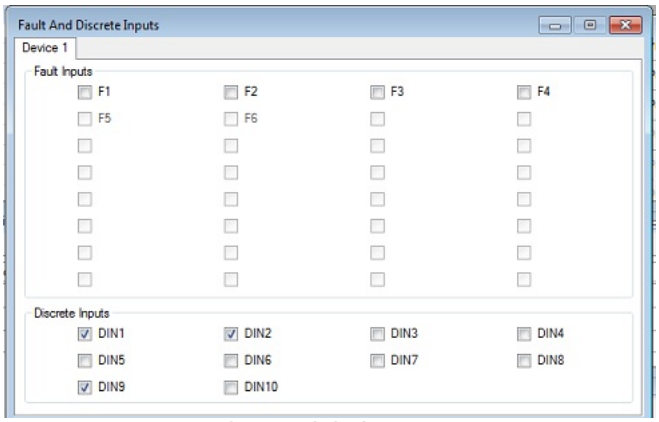


Fig.95 Digital Inputs

And we change Analog 1 = 100V and Analog 2- 500.0V. Please see Fig.96 below.

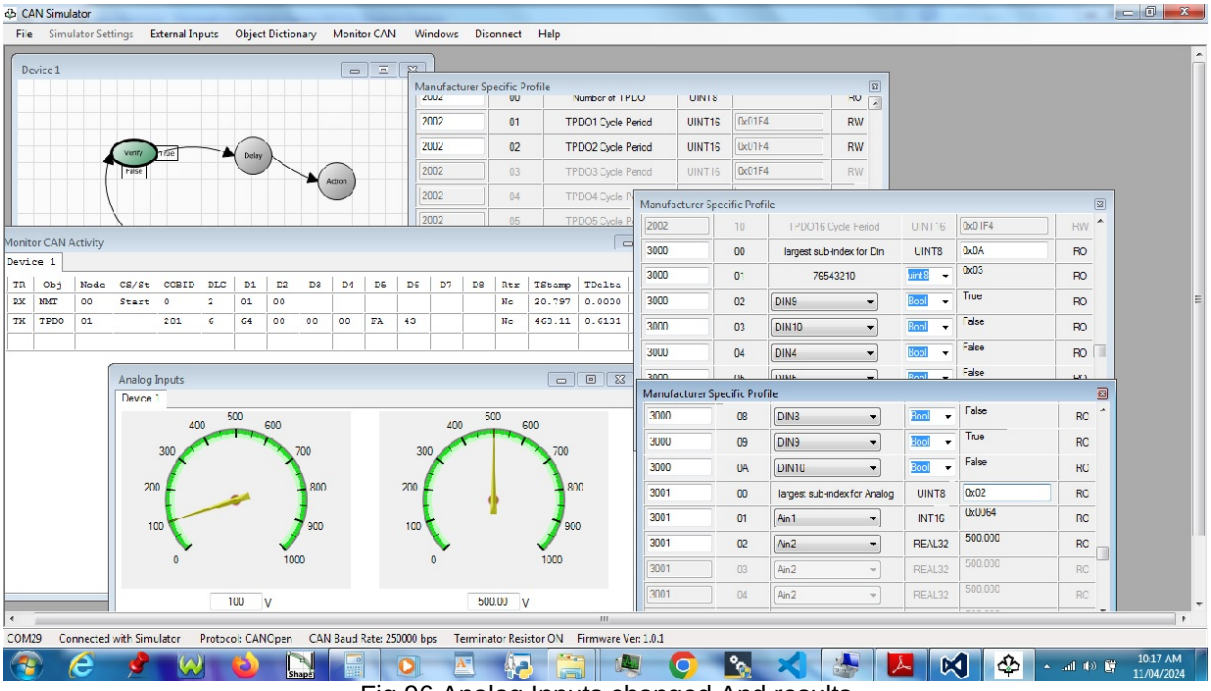


Fig.96 Analog Inputs changed And results

In Fig.96, from "Manufacturer specific Profile" windows (you can open the same dictionary windows multiple times for viewing different index), you will see Item value = 0x03 at index=3000 and subindex=1 (Because DIN1=True and DIN2=True). And you will see Item value = True at index=3000 and subindex=2 (Because DIN9=True), And you will see Item value = 0x64 at index=3001 and subindex=1 (Because Ain1=100), And you will see Item value = 500.00 at index=3001 and subindex=2 (Because Ain2=500.00),

We use Microchip CAN bus Analyzer Transmit window to sent RPDO1 with Data1=0xF8 and Data2=0

as Fig.97 Red color. Actually data=0x00F8 = Decimal 248 is TPDO1 cycle period. We will see in "Fixed Trace" window red color for TPDO1 Cycle period=0.255 in Fig.97. It is close to 0.248 Second.

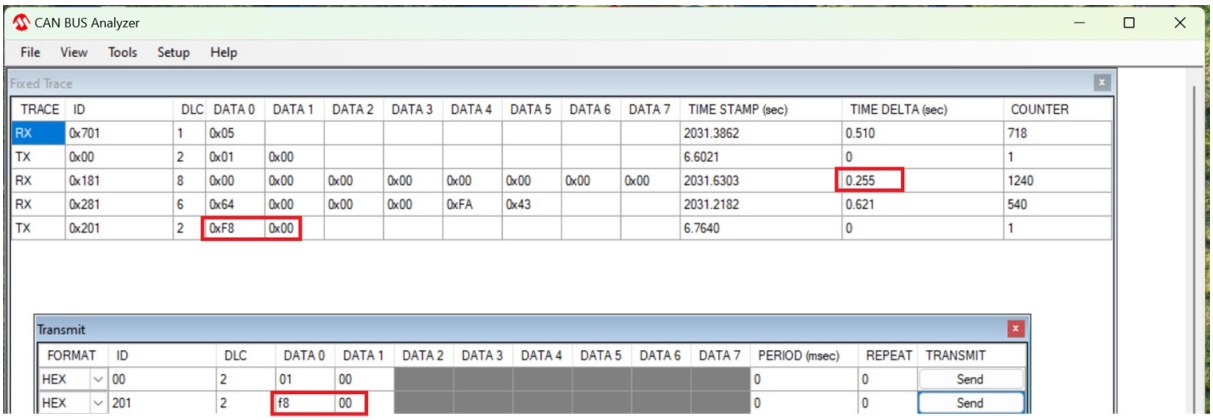


Fig.97 Send RPDO1 by CAN Bus Analyzer

We can view results in our "CAN BUS Simulator" software in Fig.98 below:

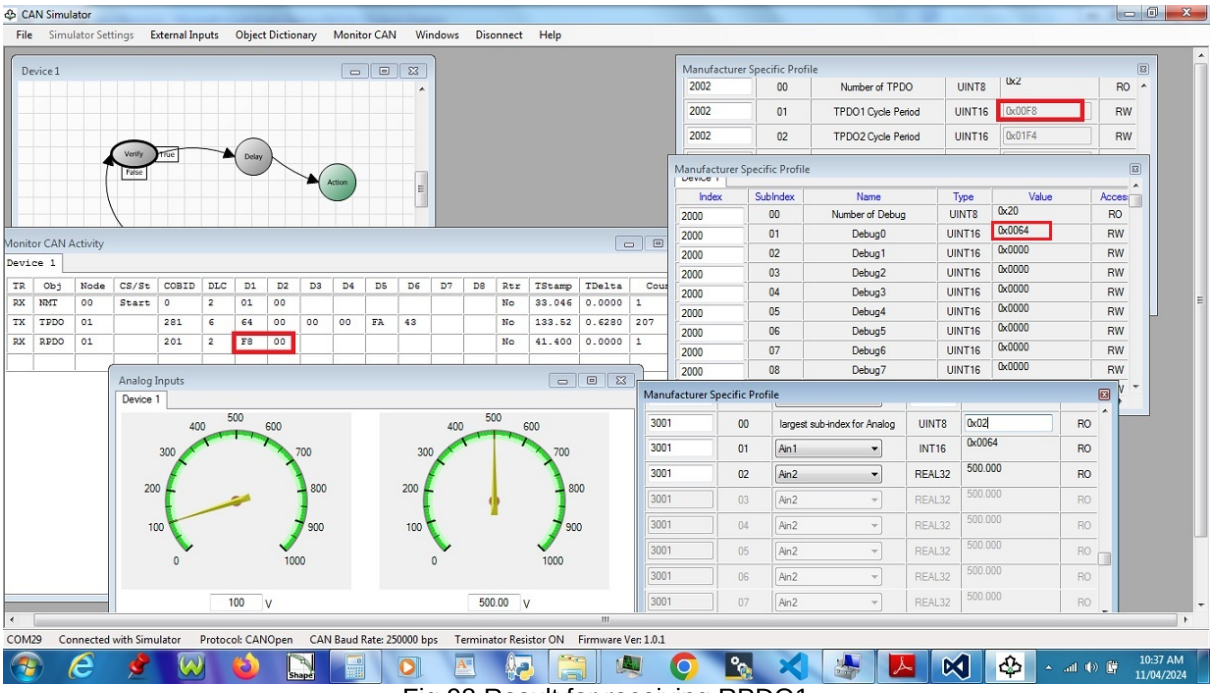


Fig.98 Result for receiving RPDO1

We saw state from "Verify" to "Delay". And after 5 sec, state change to "Action" State (Green color) in Device1 window of Fig.98. In "Monitor CAN Activity" window, please see red color, we will see receiving RPDO1 with Data1=0xF8 and Data2=0. In Fig.98, from "Manufacturer specific Profile" windows (you can open the same dictionary windows multiple times for viewing different index), you

will see Item value = 0x00F8 in red color at index=2002 and subindex=1 (because RPDO1 receives 0x00F8). And you will see Item value = 0x0064 in red color at index=2000 and subindex=1 (because it assigned Debug[0] to 100 in script of Action state).

We set F1 = true in "Fault And Discrete Inputs" window in Fig.99. In "Device1" window, we see "Fault" state becomes Red color, so fault state is activated. In "Monitor CAN Activity" window, we see "Emergency object" to send with Error code (D1 and D2)=0X1001 and Error register=0x07 (D3) in red color. In Fig.99, from "Communication Profile" window, you will see Error Register=0x07 and Error Qty=1 and standard Error field=0xA1001 (Error code plus additional information) in red color. From "Manufacturer specific Profile" window, you will see DebugSng0=10.00 in red color because DebugSng[0] is assigned to 10.00 in script of Fault State.

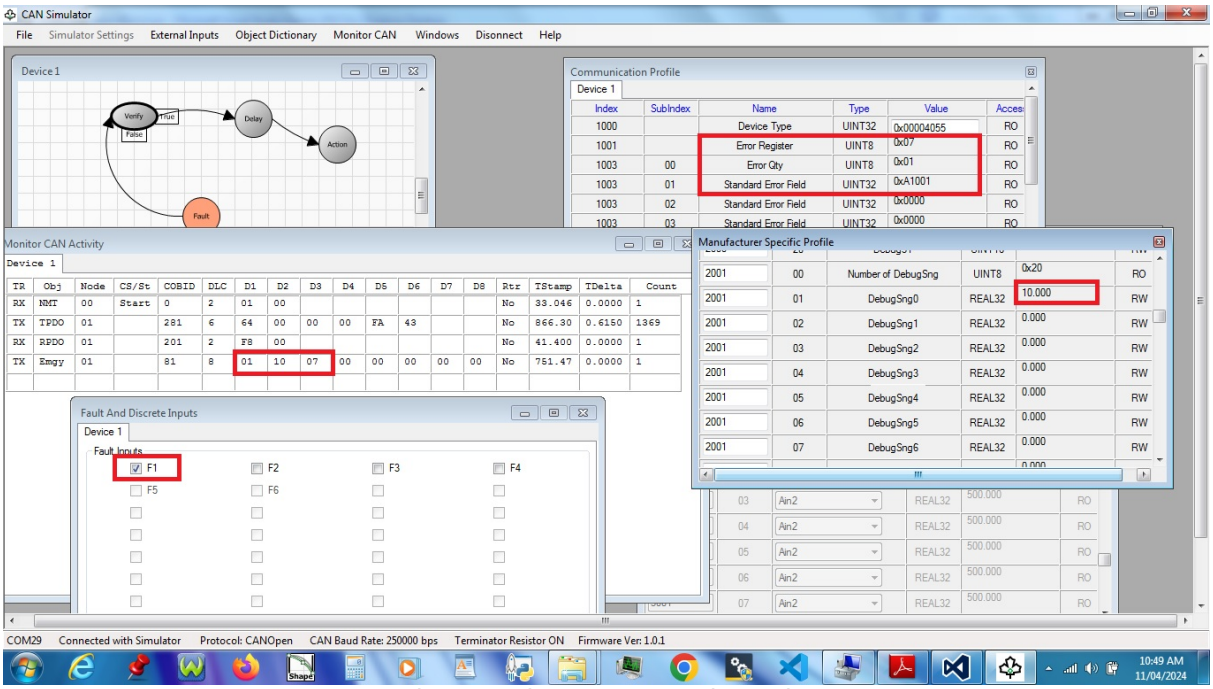


Fig.99 Fault F1 occurs and Results

This example files are in folder: Top folder\CANbusSim\Example\CANopen

8 Integrated into other software

Dafulai Electronics Provides a virtual Serial port to customers in order to customer's software can send parameter value to simulator in real time.

This virtual Serial port is .net framework component.

Firstly, let us introduce our Virtual Serial port.

Our Virtual Serial port is not truly Serial port, it cannot be controlled by Windows device Manager.

It just wrap windows pipe communication protocol in ,net frame layer. So Almost all properties and methods of regular Serial Port in .net frame component can be used in our Virtual Serial port

Our Virtual Serial port can be used in communication between different application software (process).

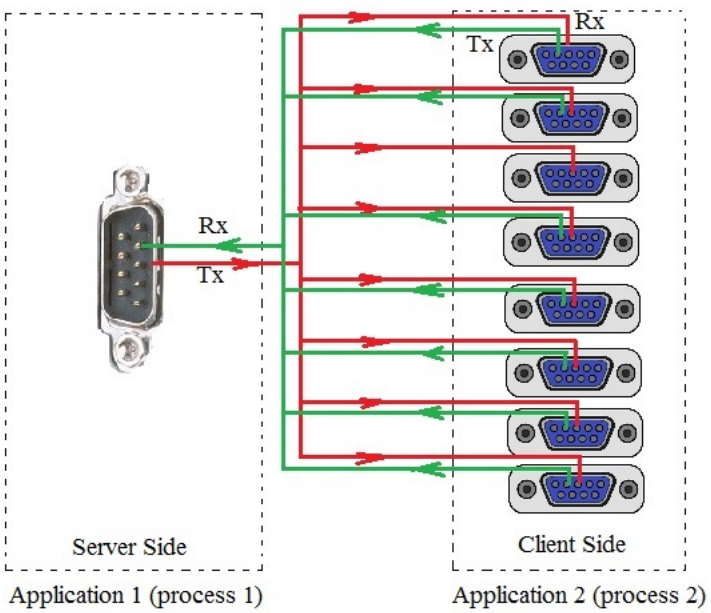


Fig. 59 Virtual Serial Port

The virtual Serial port component name is "VrSerialPort". This component function is shown in Fig 59. It is divided into "Server side" and "Client side".

"Server side" can connect only one "client" (peer to peer) , and "Server side" can connect as many as 8 "Clients". Server transmits data to all "Clients" (All clients get the same data from server). Every Client can transmit data to the same server (how to know from who? you must define data meaning, make one data as client feature such as address).

This connected "Server" and "Clients" use the same Port Name of "VirtualSerialPort". So we don't need to set up "connection" for server and clients. Server must be opened firstly. Port Name format is String "VCOMx" (x=1 ,2, 3, ...).

Compared with real Serial port component, the 2 different properties for "VrSerialPort" are "PortName" and "Side", Side value is "S" (denotes Server) or "C" (denotes Client). and further more, our "VrSerialPort" does not need baud rate.

The other write/read/close/open methods are the same as .net framework component SerialPort.

Our Simulator software has built-in "VirtualSerialPort" which is client. Customer's software will be "Server".

How to run our CAN Bus Simulator with Virtual Serial port enabled? Firstly, you must run your customized software which enable Server mode's virtual serial port which is opened.

And then in command line window, please run command:

CANSimulator VCOMx configFileName StateMachineFileName DictionaryFileName

The 1st item is Software name, you can add suffix exe, it can be CANSimulator.exe
The 2nd item is Virtual Serial Port name, it can be one of VCOM1 to VCOM12.
The 3rd item is configuration file name, you may or may not use suffix "dcfg" .This argument is option. If the 3d item missing, you cannot use the 4th item,
The 4th item is State machine file name, you may or may not use suffix "nspj" .This argument is option.
The 5th item is Dictionary file name, you may or may not use suffix "od" .This argument is option.

8.1 Communication Protocol between simulator and customer's software

We combined a group of transmitted Data for customer's software, we call it " Transmitted packet". (It stands in customer's software view point)
We combined a group of received Data for customer's software, we call it " Received packet". (It stands in customer's software view point)

For "Transmitted Data packet" , the 1st byte is "Packet type". The 2nd byte and 3rd byte are data/ command type. It is truly Data after the 3rd byte. The last 2 bytes are check sum (little Endian)
The first Byte value 0 to 3 means that this is "device input data". 0 means all data for device1, 1 means all data for device2, 2 means all data for device3, 3 means all data for device4.
The first Byte value 0xFF means that this is "Simulator Connect/Disconnect Command"

When " Transmitted packet" is "device input data", the meaning of the 2nd byte is below:

- 0x0-----Analog Input for simulator. The 3rd byte is Analog input number from 0 to 31. The 4th to 7th bytes are raw data byte for analog in little endian. The 8th and 9th bytes are check sum which is the sum of all data except checksum itself (little endian).
- 0x1-----Digital Inputs for simulator. The 3rd and the 4th bytes are all digital inputs in little endian, and Bit 0 is the 1st Digital input, Bit 1 is the 2nd Digital input, ..., Bit 9 is the 10th Digital input. The 5th and 6th bytes are check sum which is the sum of all data except checksum itself (little endian).
- 0x2-----Fault Inputs for simulator. The 3rd to the 6th bytes are all faults inputs in little endian, and Bit 0 is the 1st fault input, Bit 1 is the 2nd fault input, ..., Bit 31 is the 32th fault input. The 7th and 8th bytes are check sum which is the sum of all data except checksum itself (little endian).
- 0x3-----Warning Inputs for simulator. The 3rd to the 6th bytes are all warning inputs in little endian, and Bit 0 is the 1st warning input, Bit 1 is the 2nd warning input, ..., Bit 31 is the 32th warning input. The 7th and 8th bytes are check sum which is the sum of all data except checksum itself (little endian).

When " Transmitted packet" is "Simulator Connect/Disconnect Command", the meaning of the 2nd byte is below:

- 0xF0-----"Connect" simulator command
- 0x0F-----"Disconnect simulator command

The 3rd byte is 0x00 (must be)

The 4th and 5th bytes are check sum which is the sum of all data except checksum itself.

For " Received Data packet", the 1st byte is "Packet type", It is truly Data after the 1st byte. The last 2 bytes are check sum (little Endian)

- The first Byte value 0 to 3 means that this is "device fault/warning data output". 0 means all data for device1, 1 means all data for device2, 2 means all data for device3, 3 means all data for device4.
- The first Byte value 0xFF means that this is "All debug data output".

Let us see "device fault/warning data output" packet below:

- The 1st byte-----Device number, 0 to 3
- The 2nd to 5th bytes---------All faults signals in little endian. Bit 0 is the 1st fault output, Bit 1 is the 2nd fault output, ..., Bit 31 is the 32th fault output.
- The 6th to 9th bytes---------All Warnings signals in little endian. Bit 0 is the 1st warning output, Bit 1 is the 2nd warning output, ..., Bit 31 is the 32th warning output.
- The 10th and 11th bytes ------ check sum which is the sum of all data except checksum itself (little endian).

Let us see "All debug data output" packet below:

- The 1st byte-----0xFF
- The 2nd to 65th bytes-------All unsigned 16 bits of integer debug register value in little endian. The 2nd and 3rd bytes are for Debug(0), ..., the 64th and 65th bytes are for Debug(31)
- The 66th to 193th bytes-----All single precision float debug register value in little endian. The 66th to 69th bytes are for DebugSng(0), ..., the 190th and 193th bytes are for DebugSng(31)
- The 194th and 195th bytes------check sum which is the sum of all data except checksum itself (little endian).

8.2 An example for Integrated software

The source code for this example can be downloaded from http://dafulaielectronics.com/ThirdParty_Call_CANSimulator.zip

Firstly, you must download Virtual Serial Port component from http://dafulaielectronics.com/

VirtualSerialPort.zip

And then put Virtual Serial Port component into Toolbox of Visual Studio.

Design a form as below:

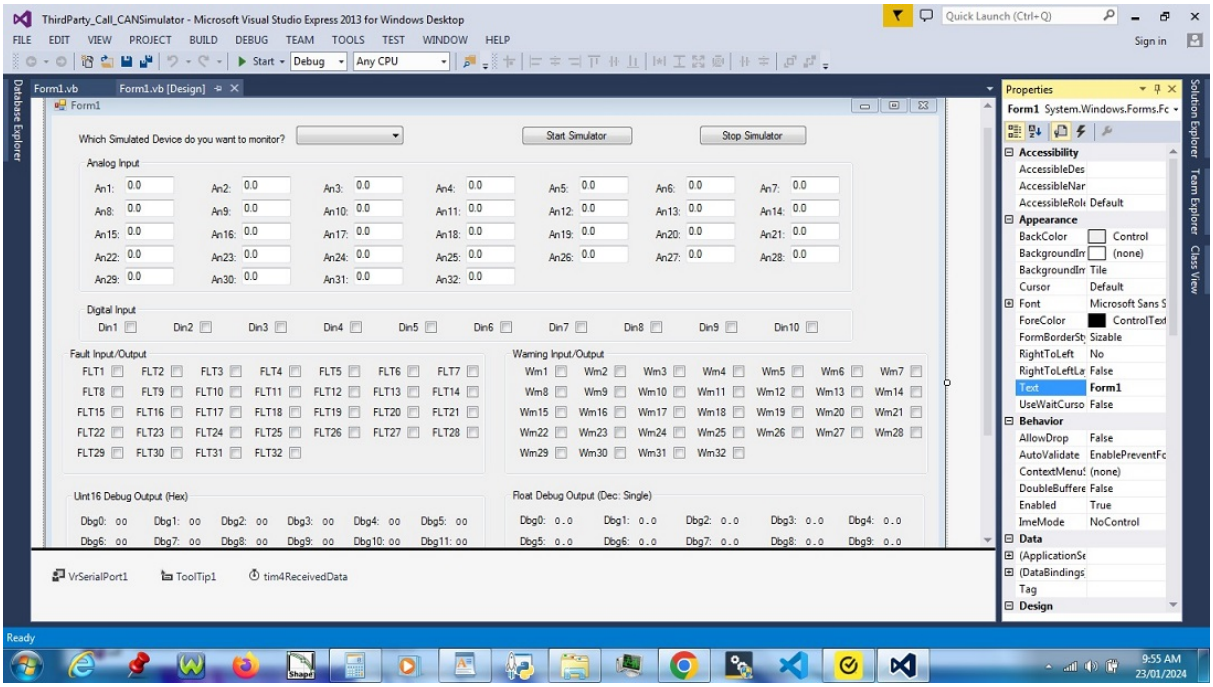


Fig.60 Example for external software to use simulator

The ComboBox is for choosing which simulated device, its name is cmbWhichDevice. Button "Start Simulator" is for connect menu in simulator, its name is "btnStart". Button "Stop Simulator" is for disconnect menu in simulator, its name is "btnStop".

The 32 analog inputs are TextBox component, its names are from txtAn1 to txtAn32. The 10 Digital Inputs are CheckBox, its names are from chkDin1 to chkDin10. The 32 Fault inputs/outputs are CheckBox, its names are chkFLT1 to chkFLT32.

The 32 warning inputs/outputs are CheckBox, its names are chkWrn1 to chkWrn32.

The 32 debug registers with 16 bits unsigned integer are Label, its names are from lblDbg0 to lblDbg31. We use hex to display its value.

The 32 debug registers with single precision float are Label, its names are from lblDbgSng0 to lblDbgSng31.

time4ReceiveData is Timer component which is for receiving simulator Data. The time interval is 200ms.

The virtual Serial Port name is "VrSerialPort1". The property of Side is "S" .

For our class Form1, we put the following structure definitions:

```
Structure OutAnalogInforStru
   Dim DeviceNo As Byte  '0, 1, 2, 3 are deviceNo value
   Dim WhichType As Byte  '0 ----Analog, 1----Din, 2----Fault In,
       3-----WarnIn, it should be 0
   Dim AnalogNo As Byte   '0 to 31
   Dim Analog As Single  '32 analog,  Physical Value
   Dim checksum As UShort
End Structure
Structure OutDinInforStru
   Dim DeviceNo As Byte  '0, 1, 2, 3 are deviceNo value
   Dim WhichType As Byte  '0 ----Analog, 1----Din, 2----Fault In,
       3-----WarnIn, it should be 1
   Dim Din As UShort '10 din
   Dim checksum As UShort
End Structure
Structure OutFaultInforStru
   Dim DeviceNo As Byte  '0, 1, 2, 3 are deviceNo value
   Dim WhichType As Byte  '0 ----Analog, 1----Din, 2----Fault In,
       3-----WarnIn, it should be 2
   Dim FaultIn As UInteger '32 fault inputs
   Dim checksum As UShort
End Structure
Structure OutWarnInforStru
   Dim DeviceNo As Byte  '0, 1, 2, 3 are deviceNo value
   Dim WhichType As Byte  '0 ----Analog, 1----Din, 2----Fault In,
       3-----WarnIn,it should be 3
   Dim WarnIn As UInteger '32 fault inputs
   Dim checksum As UShort
End Structure
Structure OutCMDStru
   Dim DeviceNo As Byte  '0xFF are deviceNo value--Command
   Dim Command As UShort   'Command : First byte 0x0-None, 0xF0--connect,
       0x0F--Disconnect
       ' The 2nd byte:  No meaning
   Dim checksum As UShort
End Structure
Structure InDataInforStru
   Dim DeviceNo As Byte  '0, 1, 2, 3 are deviceNo value
   Dim FaultIn As UInteger '32 fault inputs
   Dim WarnIn As UInteger '32 fault inputs
   Dim checksum As UShort
End Structure
Structure InDebugInforStru
   Dim DeviceNo As Byte  '0xFF are Debug infor Flag
   <VBFixedArray(31)> Dim Debug() As UShort  '32 double bytes of Debug
   <VBFixedArray(31)> Dim DebugSng() As Single  '32 four bytes of DebugSng
   Dim checksum As UShort
End Structure
```

We put the following variables for class Form1:

```
    Dim AnalogToSimulator As OutAnalogInforStru
    Dim DinToSimulator As OutDinInforStru
    Dim FaultToSimulator As OutFaultInforStru
    Dim WarnToSimulator As OutWarnInforStru
    Dim CMDToSimulator As OutCMDStru
    Dim DataFrmSimulator As InDataInforStru
    Dim DataFrmSimulator_old As InDataInforStru
    Dim DebugFrmSimulator As InDebugInforStru
    Dim DebugFrmSimulator_old As InDebugInforStru
    Dim SyncLockReceved As String = ""

    Private MySimulatorInstance As New Process()
```

We make initialization in Form1 Load event below:

```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
   cmbWhichDevice.SelectedIndex = 0
   VrSerialPort1.PortName = "VCOM2"
   Try
      VrSerialPort1.Open()
   Catch ex As Exception
      MsgBox(ex.Message)
      Application.Exit()
   End Try

   Environment.CurrentDirectory = "D:\Project\CANBus_Simulator"
   MySimulatorInstance.StartInfo = New ProcessStartInfo("CANSimulator.exe", _
    "VCOM2  ""C:\Users\test_can\test2"" ""C:\Users\test_can\test7"" ")
   MySimulatorInstance.Start()

   DataFrmSimulator.DeviceNo = 0
   DataFrmSimulator.FaultIn = 0
   DataFrmSimulator.WarnIn = 0
   DataFrmSimulator_old.DeviceNo = 0
   DataFrmSimulator_old.FaultIn = 0
   DataFrmSimulator_old.WarnIn = 0

   DebugFrmSimulator.Debug = New UShort(31) {}
   DebugFrmSimulator_old.Debug = New UShort(31) {}
   DebugFrmSimulator.DebugSng = New Single(31) {}
   DebugFrmSimulator_old.DebugSng = New Single(31) {}

End Sub
```

In above code, for Environment.CurrentDirectory, you must use the directory name of your file

"CANSimulator.exe" instead of `"D:\Project\CANBus_Simulator"` which is location of our file "CANSimulator.exe".

For Statement `"MySimulatorInstance.StartInfo = New ProcessStartInfo(.....)"`, the `"VCOM2"` of the second parameter `"VCOM2 ""C:\Users\test_can\test2"" ""C:\Users\test_can\test7"" "` can be changed by any one of "VCOM1 to VCOM12". However, it must be equal to the PortName in Statement of `"VrSerialPort1.PortName= ..."`.

`""C:\Users\test_can\test2""` must be replaced by your configure file. The 2 double quotation marks denote one double quotation mark.

`""C:\Users\test_can\test7""` must be replaced by your state machine file.

You can directly use our example code after above changes according to your file locations.

9 Notice

IMPORTANT NOTICE

The information in this manual is subject to change without notice.

Dafulai's products are not authorized for use as critical components in life support devices or systems. Life support devices or systems are those which are intended to support or sustain life and whose failure to perform can be reasonably expected to result in a significant injury or death to the user. Critical components are those whose failure to perform can be reasonably expected to cause failure of a life support device or system or affect its safety or effectiveness.

COPYRIGHT